Bolt Beranek and Newman Inc.

bbn

AD-A134 102

6

Report No. 5421

Research In Knowledge Representation For Natural Language Understanding

Annual Report
1 September 1982 to 31 August 1983

S ELECTE OCT 27 1983 D

A

Prepared for:
Defense Advanced Research Projects Agency

This document has been approved
for public release and sale; its
distribution is unlimited.

DTIC FILE COPY

83 10 27 011

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
BBN Report No. 5421

2. GOVT ACCESSION NO.
AD-A134 602

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING
Annual Report
1 September 1982 - 31 August 1983

5. TYPE OF REPORT & PERIOD COVERED
Annual Report
9/1/82 - 8/31/83

6. PERFORMING ORG. REPORT NUMBER
BBN Report No. 5421

7. AUTHOR(s)
Sidner, C., Bates, M., Bobrow, R., Goodman, B. Haas, A., Ingria, R., Israel, D., McAllester, D. Moser, M., Schmolze, J., Vilain, M.

8. CONTRACT OR GRANT NUMBER(s)
N00014-77-C-0378

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Bolt Beranek and Newman Inc.
10 Moulton St.
Cambridge, MA 02238

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

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
Dept. of the Navy
Arlington, VA 22217

12. REPORT DATE
October 1983

13. NUMBER OF PAGES
234

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

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.

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

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Artificial intelligence, natural language understanding, knowledge representation, semantics, semantic networks, KL-ONE, NIKL, belief and knowledge, reasoning, RUP, syntax, parsing, noun phrase interpretation, reference, miscommunication.

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

BBN's ARPA project in Knowledge Representation for Natural Language Understanding is aimed at developing techniques for rendering computer-based assistance to a decision maker who is attempting to understand and react to a complex, evolving system or situation. The decision maker's access to the situation is mediated by an intelligent graphics display system, which is controlled largely through natural language input. The work during this past year falls into two main categories: fundamental

(cont'd...)

Unclassified.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Cn'y

20. Abstract (cont'd.)

problems in knowledge representation and reasoning, and fluent natural language understanding. In this report, we first give a brief overview and summary of the activities of the project during the year. This is followed by a series of detailed presentations of research in particular areas. In addition, we document publications and presentations by members of the research group.

Report No. 5421 Bolt Beranek and Newman Inc.

RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING

Annual Report
1 September 1982 to 31 August 1983

October 1983

DTIC
COPY
INSPECTED
2

Accession For
NTIS GRA&I ☒
DTIC TAB □
Unannounced □
Justification

By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

Prepared for:

Defense 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 1984

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 uF CONTENTS

1. INTRODUCTION

BBN's ARPA project in Knowledge Representation for Natural Language Understanding is aimed at developing techniques for rendering computer-based assistance to a decision maker who is attempting to understand and react to a complex, evolving system or situation. The decision maker's access to the situation is mediated by an intelligent graphics display system, which is controlled largely through natural language input. A typical and motivating instance is that of a military commander in a command and control context, either of strategic situation assessment or of more tactical crisis management. In such situations the commander requires a flexible and easily controllable system capable of manipulating large amounts of data and, most importantly, of presenting information in a variety of forms suited to the user's expressed or inferable needs and capacities.

A display system of the kind envisaged would have the capacity to present information in tabular, graphical, textual, and perhaps cartographic forms. The user of such a system must be able to monitor, add, change and delete information and, independently, to create and alter the various representational forms. Moreover, for the system to be truly flexible and adaptive, it must maintain models of the domain (situation) being represented, of the representational systems at its disposal, and of the user's conceptions of these domains, situations, and systems of representation. For this last purpose, the system must also be able to construct models of its interactions with the user.

On the basis of these different kinds and sources of information, the system must produce intelligible and appropriate displays in response to high-level descriptions and commands. That is, the commander can usually be expected to request a presentation of certain aspects of the situation or system being monitored in terms appropriate to the domain itself and not in terms of display forms. Even when the request, explicit or implicit, is expressed in terms of display forms, the specification will typically be at a level of abstraction appropriate to the commander's purpose - not to those of a graphics system designer or programmer. The system must be able to accept a description of the information to be represented together with an abstract specification of a display-type and then it must intelligently determine the details required actually to produce an effective dsiplay. Finally, given information about the user's knowledge of the situation being monitored and his particular concerns with respect to it, the system must, in some cases, be able to infer what kind of display a user might want to see, produce it and monitor the user's response to its initiatives.

The crucial requirements for a medium of communication with such a system are robustness, flexibility, the ability to express specifications while abstracting from details of various kinds, and the ability to express conceptualizations of both the presented domain and the modes of display in ways that match a user's conceptualization. By far the most natural form of access to and control over such a system for most users will be through the use of natural language input. Hence a major focus of our research has been the design of a system powerful enough to represent the content of natural language utterances together with facts about the user's beliefs and goals as these are communicated in the user's interactions with the system. Such a representational formalism must also express, in usable form, information about the domain or situation being monitored and the nature of the display system itself.

The development of the KL-ONE knowledge representation system has been a significant result of this aspect of our effort. KL-ONE has generated a great deal of interest in the research community and has been adopted by a number of groups. It has been transported to a variety of architectures and programming languages; versions of it have been implemented in SmallTalk, on various Xerox processors, and in FranzLisp for the DEC VAX series and other DEC machines. The most complete implementations are in InterLISP on the PDP-20, the BBN Jericho and the Xerox Dolphin.

Over the last two years, the development of KL-ONE has been greatly enriched by cooperation with the ARPA sponsored Consul group at ISI. Indeed, the major activity this year in knowledge representation has been the re-design and reimplementation of KL-ONE, a joint activity of the BBN and ISI groups. This effort has resulted in NIKL - a New Implementation of KL-ONE. NIKL represents a significant streamlining and simplifying of the KL-ONE system - without significant loss of expressive power. The system has been, as hoped, much easier to comprehend, to implement and to debug. Moreover, there is a simple enough mapping between KL-ONE constructs and NIKL constructs to guarantee continued applicability of the descriptions and analyses of the KL-ONE system and to allow us to continue using essentially the same graphical notation for publication.

The first two papers in this report are devoted to the knowledge representation system. The first presents an introduction to and overview of the new KL-ONE system; the second contains a semantic account of the core of the system and, based on that account, a description and analysis of the classification algorithm. These are followed by a short description of the facility for automatically drawing KL-ONE networks.

One crucial feature of NIKL is that it was designed, and is being implemented, explicitly with the intent that it be interfaced with an assertional formalism which includes data structures for propositions, predicates and terms and algorithms that realize powerful, but controllable, inference procedures over those structures. As a first step, we have chosen to connect NIKL with RUP (Reasoning Utility Package), a truth maintenance system developed by David McAllester of MIT. This decision has meant that a number of KL-ONE constructs which were designed specifically to handle assertional phenomena have disappeared from NIKL, only to reappear in RUP - or, more accurately, in the NIKL-RUP system. The fourth paper is a description and analysis of the RUP-NIKL interface.

One of the tasks of a system of the kind envisaged is the maintenace of a model of the user, in particular of his/her beliefs about the situation being monitored, about the display system itself and about the history and current state of his/her interactions with that system. The problems that arise in representing the mental states (beliefs and desires) of agents in ways that can be used to predict and explain their actions are highlighted in a situation in which the user is presumed to be communicating his beliefs and desires to a computational artifact through the medium of a high-level, implementation independent language. The fifth section of the report presents an attempt at a uniform treatment of a large number of these issues, a treatment based on the idea that sentences in a formal language are useful representatives of the contents of agents' mental states.

The work on knowledge representation has been motivated by the task of designing an intelligent computational assistant to a decision-maker who communicates with the system in English. An essential component of the overall system is a grammar-parser formalism adequate to a rich fragment of English and able to produce appropriate structures on which to base semantic interpretation. The RUS parser and the PSI-KLONE (now PSI-NIKL) system for semantic interpretation afford us these tools. RUS, like KL-ONE, has been adopted by a number of research projects with a wide variety of domains. A major effort this past year has been directed at making RUS more accessible to users and investigating its application in other contexts. A significant result of this work has been the development of IRUS (Information Retrieval Using RUS), a natural language interface transportable to a variety of database systems. The seventh chapter in this report contains a description and analysis of IRUS, and the eighth provides an update on the status of the basic RUS system.

In any natural language understanding system, a crucial issue is the treatment of the different kinds and uses of referring expressions. The ninth section of the report presents a systematic analysis of certain central uses of both definite and indefinite noun phrases. One phenomenon treated in this analysis is that not all uses of referring expressions are intended to direct or even allow the hearer to fix on a particular object as the referent of the expression. The tenth section of the report analyzes a range of cases in which *that* is the speaker's intent but in which, for a variety of reasons, the hearer is unable to fix on the object that the speaker has in mind. A system for handling such cases of miscommunication is described and motivated.

The last section of the report contains a list of publications in which other aspects of our research over the past year are presented.

2. AN OVERVIEW OF NIKL, THE NEW IMPLEMENTATION OF KL-ONE

M. G. Moser

KL-ONE is a tool for forming conceptual descriptions, allowing the system using it to construct a knowledge base representing the beliefs of a reasoning entity. New descriptions can be formed with a small number of operations to combine and relate those already in the knowledge base or they can be introduced as primitves. This paper is an overview of the structures and how they interrelate in the latest implementation of KL-ONE, New Implementation of KL-ONE or NIKL.

There are several interfaces to KL-ONE for building a knowledge base, or KL-ONE net. One of these, JARGON, was developed to investigate the idea that certain English syntactic structures are natural expressions of the semantics of KL-ONE structures. JARGON is only partially implemented, and its language will be used in this overview to clarify examples. The complete interface, CKLONE, allows users to define, name, update, and file KL-ONE Concepts using LISP forms. Typically, CKLONE is used to build a KL-ONE network for a particular domain and the domain application system will access the network using PENNI, a separate assertional language.

A KL-ONE Concept is depicted with an ellipse to which depictions of the various structures used to describe that Concept are attached. Each Concept represents a class of things in the world of concern. We usually name a Concept after the elements in the set it denotes. Figure 1 shows a Concept

FIG. 1. *|C|ANIMAL*

named "ANIMAL" that denotes the set of all animals. |C|concept-name is used to denote a KL-ONE Concept structure we have named "concept-name".

Every KL-ONE network includes a Concept that is defined to represent the super-class of the classes denoted by all other Concepts. We call this Concept "C-TOP", or sometimes "THING".

If we simply assert that there is some Concept called ANIMAL without describing this Concept further, we have established some sub class cf the class represented by |C|THING, as shown in Figure 2.

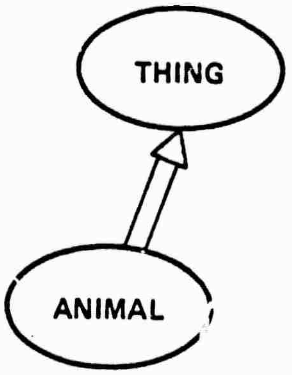


FIG. 2. *AN ANIMAL IS A THING.*

The arrow in Figure 2 expresses that |C|THING subsumes, or is a SuperC of, |C|ANIMAL and that |C|ANIMAL specializes, or is a SubC of, |C|THING. When one Concept is a SubC of another, it denotes a subclass of the class denoted by its parent. In terms of a frame system, a Concept would correspond to a frame and its relationship to its subsumers would correspond to an is-a link. A SubC will inherit all the components of its parents and will have one essential component that distinguishes it as a specialization of its parent. (We have not yet established the specializing component for |C|ANIMAL.)

The collection of Concepts is organized into a taxonomic net (See Figure 3). Adding a concept to the net requires installing it in the taxonomy. Because Concepts derive much of their meaning from their SuperCs, it is crucial that each Concept be installed under the most specific Concepts possible and subsuming the most general Concepts possible. This is the job of the KL-ONE classification operation.

There is an important distinction to be made between definitions of terms and statements about things represented by those definitions. Our knowledge base is definitional. Each Concept represents some class of things we want to make assertions about. It may be thought of as a logical predicate, a complete specification of the necessary and sufficient conditions for membership in the class represented. Each member of this class is called an instance of the Concept. Concepts are definitional while instances are assertional.

For systems to use an existing KL-ONE network, there is a companion assertional system for KL-ONE, PENNI. PENNI reasons about individuals (i.e., instances of Concepts) of interest to the current domain application.

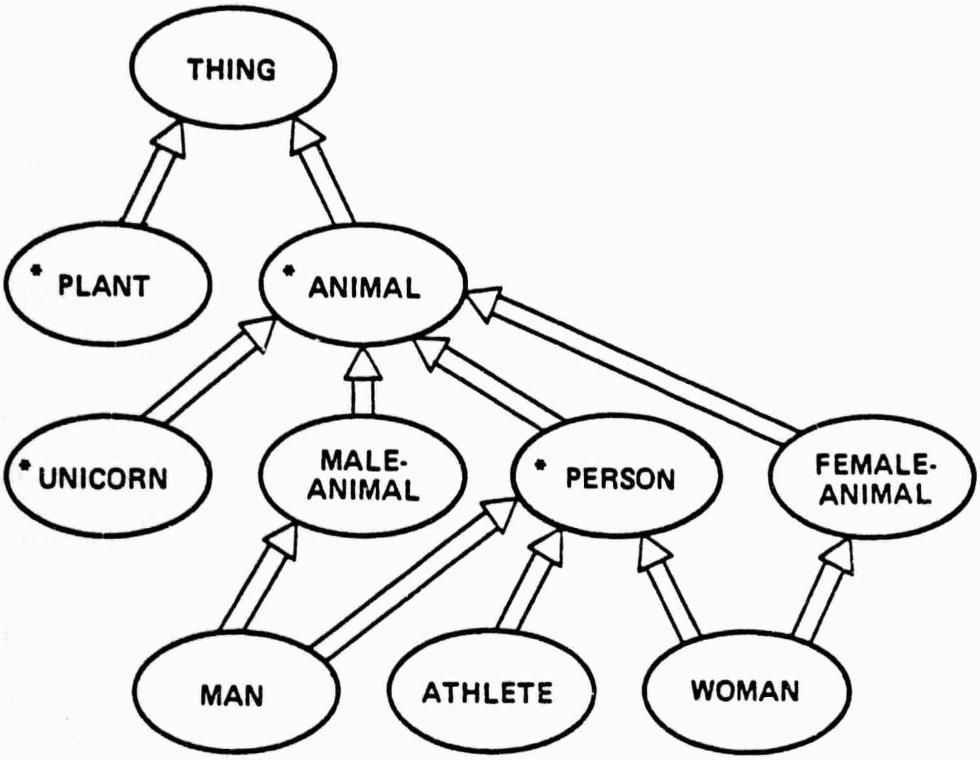


FIG. 3. *An example KL-ONE taxonomy*

Assertions are made in terms of the structures in the KL-ONE network, and consistency among assertions is maintained.

There is a class of terms we would like to be able to represent in our knowledge base but which cannot be fully defined. Natural kind terms, such as person or elephant, can only be described. Such Concepts can be created in KL-ONE and their relevant properties described, but a PrimitiveClass marking must be assigned to indicate that there is some distinction between the new Concept and its subsumers which is not defined within the knowledge base. A PrimitiveClass marking for the Concepts in the ANIMAL taxonomy of Figure 3 is indicated with a "*". The KL-ONE components which could specify the Concepts not marked with a "*" will be introduced shortly.

This PrimitiveClass is an important component of Concepts whose KL-ONE structure provides necessary but not sufficient criteria for membership in the class they are representing. Marking a Concept with a PrimitiveClass indicates that membership in the class represented must be established in some way external to KL-ONE. Other Concepts will not be subsumed by a Primitive Concept unless explicitly stated. This allows natural kind Concepts to be included in the knowledge base and to be treated like defined Concepts.

The classifier is only concerned with the presence or absence of a PrimitiveClass, the value and structure of the PrimitiveClass marking can be anything the system choses. It may contain information for the system to use to determine if it should explicitly state something is subsumed by the Concept, such as a function name.

Some Concept descriptions are so specific they have only one instance in our world of interest. We think of these as Individual Concepts, while those with multiple instances are thought of as Generic Concepts. Although the distinction between individual and generic Concepts is assertional in nature, KL-ONE allows the system to mark a Concept as being individual. This marking is for the convenience of a system accessing a KL-ONE network and is ignored by KL-ONE.

KL-ONE nets are built incrementally. We can not expect that the knowledge base be coherently and systematically described in its entirity at one time. There is an infinite number of conceptual specifications possible, and the current taxonomy is a small selection from them. Relevant generalizations and specializations of Concepts will become apparent both as the network is being built and as it is being accessed by PENNI. For instance, it might be useful to define |C|LIVING-BEING which would subsume |C|ANIMAL and |C|PLANT in our example, and |C|MAMMAL which would be between |C|ANIMAL and |C|PERSON. These new Concepts could be added as the network was being built or as it was being used.

Since our taxonomy must evolve independently of the order in which Concepts are described, we must be able to install new Concepts anywhere in the taxonomy, i.e. to classify Concepts. A new Concept is described by building a ConceptSpec which is then installed in the taxonomy by the classification operation. Once installed, no more components may be added to that Concept, although it may be generalized and specialized by the classification of other ConceptSpecs.

In addition, there are several components which express interesting things but which are not used in classification. That is, any inferences that could be made on the basis of these components are not used by the classifier. These may be added to a Concept description before or after it is installed because they will not affect a Concept's place in the taxonomy. These things include Individual marking, explained above, and DisjointnessClass, Covering, Data, IData, and InverseRole, all to be discussed later.

The names we choose are only convienent labels for our Concept descriptions. KL-ONE maintains the structure of the network, leaving the interpretation of that structure to the system accessing it.

Roles are KL-ONE entities which represent logical associations between Concepts. A Role describes an aspect or property of one Concept by relating it to another Concept. As the functional notation in Figure 4 illustrates, a Role is formally a two place relation whose domain and range are represented by Concepts. To expand the frame analogy, the slots of a frame would be similar to Roles.

A Role maps each instance of the domain Concept into a set of instances of the range Concept. An instance of the range Concept is called a filler of the Role for the appropriate instance of the domain Concept. A Role actually represents a set of fillers for each instance of the domain Concept.

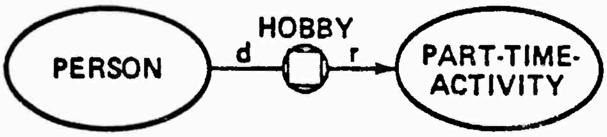


FIG. 4. *A PERSON HAS PL[1] HOBBY WHICH ARE PL PART-TIME-ACTIVITY.*

PART-TIME-ACTIVITY = HOBBY(PERSON)

A KL-ONE Role is depicted with a square enclosed by a circle. Figure 4 denotes a relation, called "HOBBY", which, for every instance of |C|PERSON describes a set of instances of |C|PART-TIME-ACTIVITY. Roles, like Concepts, are labeled by the network builder to indicate what the structure represents. |R|role-name is used to denote a KL-ONE Role structure we have named "role-name".

[1] Nouns are pluralized in JARGON by using the special morpheme "PL".

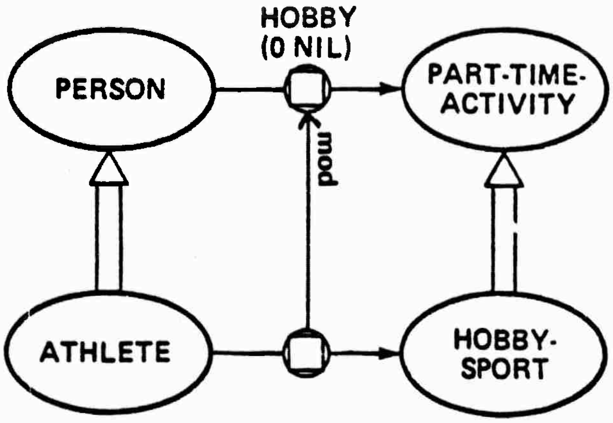


FIG. 5. *A PERSON WHOSE PL HOBBY ARE PL HOBBY-SPORT IS CALLED AN ATHLETE.*

A Role is associated with, or attached to, its domain Concept, the most general Concept at which the Role makes sense. Because the relation denoted by a Role will also apply to all subclasses of its domain, Role attachment is inherited by all SubCs of the domain Concept. In fact, one way to define a SubC of the domain is to describe a more specific range of an inherited Role. In Figure 5, we have defined the Concept of an amateur athelete, |C|ATHELETE, by restricting the range of |R|HOBBY to |C|HOBBY-SPORT. This is expressed with a RoleRestriction. This definition of an athelete does not include people who have additional hobbies which are not sports. We will give a better definition shortly.

Every attachment of a Role tc a Concept has an associated RoleRestriction which describes the range of the relation denoted by the Role when the domain is restricted to the class denoted by the Concept. A Concept inherits both the Roles and the RoleRestrictions of its SuperCs. A RoleRestriction has two elements, a value restriction, the VR, and a number restriction, the #R.

When the domain of a relation is restricted to a Concept, the VR of the Role at that Concept is the Concept denoting the corresonding class for the range. This will be either the inherited VR or a SubC of the inherited VR. In the example in Figure 5, the VR of |R|HOBBY at |C|ATHLETE is |C|HOBBY-SPORT.

The #R is a number range of the form (Min Max) that indicates how many members may be in the set of fillers of the Role for an instance of the Concept. This will either be the inherited #R, or a more restricted #R. Figure 6 shows an example of defining a Concept using its #R on a Role.

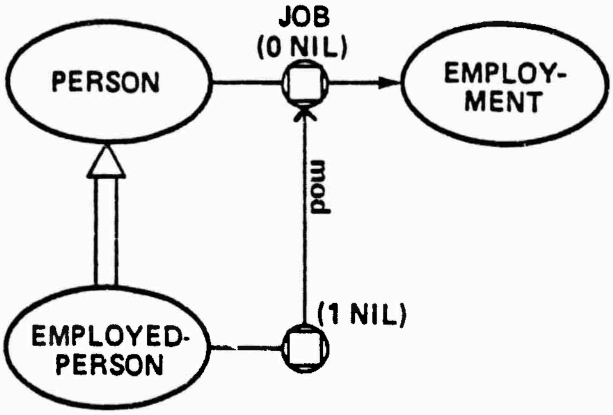


FIG. 6. *AN EMPLOYED-PERSON IS A PERSON WHO HAS AT LEAST ONE JOB.*

Through their function of relating two Concepts, Roles describe the essential properties that allow us to distinguish SubCs. By tightening the inherited RoleRestriction of |R|HOBBY, we defined two SubCs of |C|PERSON.

The structure of a Role includes the Concepts representing the domain and range of the relation denoted by that Role, PrimitiveClass (just as for

Concepts), and all its SuperRoles. A RoleRestriction is a component of a Concept definition and is independent of the restricted Role's specification.

The domain of a Role is the most general Concept at which that Role could make sense. A domain is a component of a Role specification and is independent of the domain Concept's specification. Usually, but not always, a Concept that is the domain of a Role has no RoleRestriction for that Role and so its place in the taxonomy is not affected by it. Absence of a RoleRestriction is equivalent to a RoleRestriction whose VR is the range of that Role and whose #R is zero to infinity, written (0 NIL).

We define subsumption among Roles just as we did for Concepts. In other words, we can express that one Role denotes a more specific relation than that denoted by another Role. Every KL-ONE net has a Role which is defined to represent the most general relation of the relations denoted by all other Roles. We call this Role "R-TOP", or "RELATION". Its domain and range are |C|THING, as shown in Figure 7.

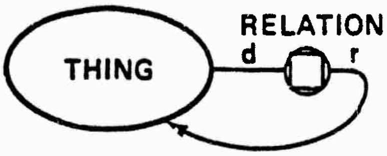


FIG. 7. *|R|RELATION*

Roles have their own taxonomy with many parallels to the Concept taxonomy. Some of the other Roles which describe our example taxonomy are shown in Figure 8.

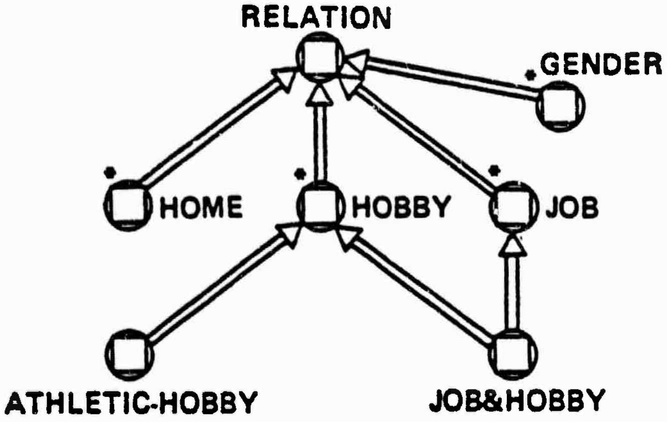


FIG. 8. *An example Ro.e taxonomy*

The subsumption arrow in Figure 8 indicates that |R|ATHLETIC-HOBBY differentiates, or is a SubR of, |R|HOBBY, and that |R|HOBBY subsumes or is a SuperR of |R|ATHL.ETIC-HOBBY.

The specification of a Role consists of the conjunction of the specifications of its SuperRs plus an essential distinction. This essential distinction may be a PrimitiveClass, a more limited range, or multiple SuperRs. A RoleSpec is fully described and then classified, like a ConceptSpec. A classified Role's domain will be the conjunction of the domains of its RoleSpec and all its SuperRs.

Just as with our Concept taxonomy, we need a classification operation that allows us to generalize and differentiate Roles whenever the need arises so that their place in the taxonomy is independent of the order in which they are installed. There are several Role descriptors for the convenience of a network user, not used in classification, which will be explained later.

To return to our |C|ATHLETE example, suppose we have established the ConceptSpec and RoleSpec shown in Figure 4.

If we establish the subset of each person's hobbies which are sports, Figure 9, then an amateur athlete is someone who has at least one member in this subset, Figure 10.

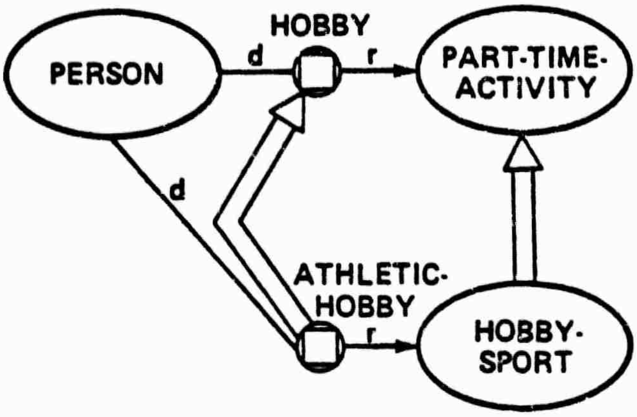


FIG. 9. *SOME OF A PERSON 'S PL HOBBY ARE CALLED PL ATHLETIC-HOBBY WHICH ARE PL HOBBY-SPORT.*

Another component of Concepts, RoleConstraints, represents a relationship between the sets of fillers of Roles at that Concept. A RoleConstraint is defined at its enclosing Concept and is inherited by that Concept's SubCs. As shown in Figure 11, the introduction of a RoleConstraint is another way to describe the specialization of a Concept.

A RoleConstraint consists of a constraint type which is either EQUAL, SUPERSET, or SUBSET and two RoleChains which are lists of Roles. The first Role in the list must be attached to the enclosing Concept of the

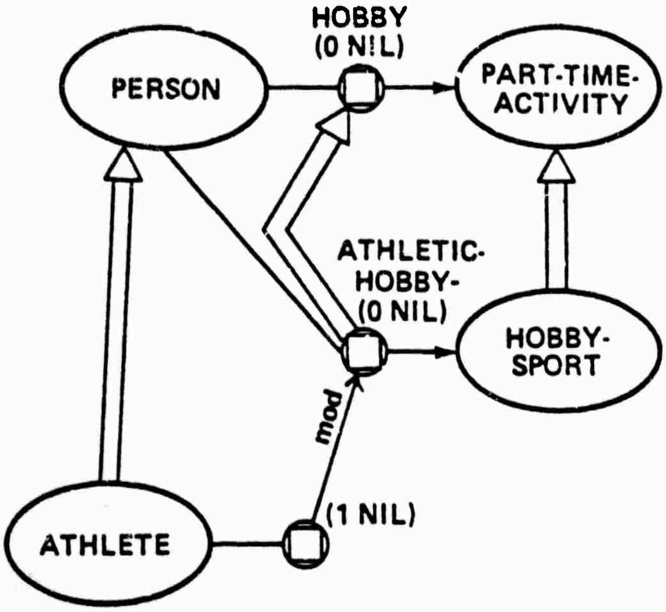


FIG. 10. *AN ATHLETE IS A PERSON WHO HAS AT LEAST ONE ATHLETIC—HOBBY.*

RoleConstraint. Each subsequent Role must be attached to the VR in the RoleRestriction at the Concept where the previous Role in the list was attached.

In effect, each RoleChain represents a composite relation, composed of the relations represented by the Roles in the chain, with its domain as the enclosing Concept. In Figure 11, the functional notation for Roles illustrates the composite relation represented by the RoleChains. The constraint type establishes, for any instance of the enclosing Concept, the relation between the instances of the two composite relations.

We are now in a position to give a more concise explanation of how one Concept or Role subsumes another. For simplicity, the terms Object and

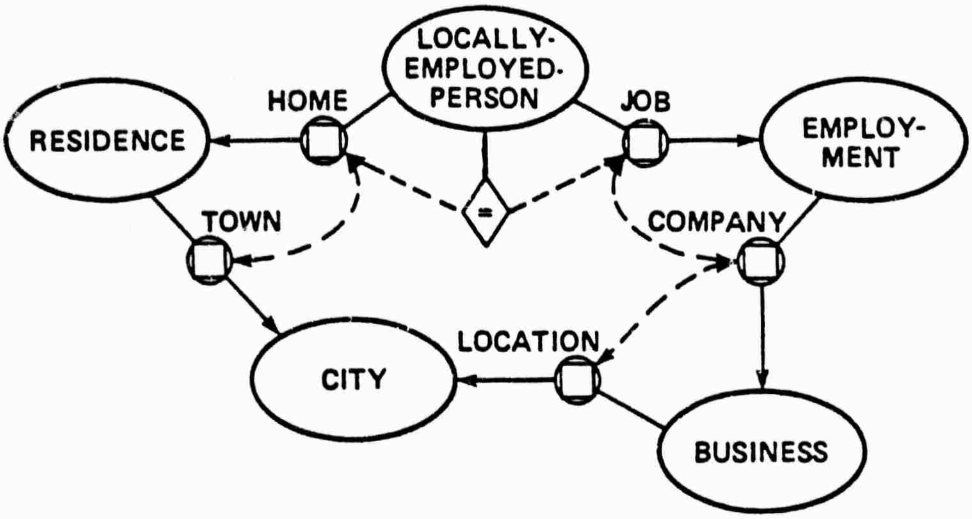


FIG. 11. *A LOCALLY-EMPLOYED-PERSON IS AN EMPLOYED-PERSON. THE LOCATION OF THE COMPANY OF THE JOB OF A LOCALLY-EMPLOYED-PERSON IS THE SAME AS HER HOME 'S TOWN.*

LOCATION (COMPANY (JOB (LOCALLY-EMPLOYED-PERSON))) = TOWN (HOME (LOCALLY-EMPLOYED-PERSON))

ObjectSpec will be used for explanations which can apply to Concepts or Roles. Subsumption means that the subsumee represents a subset of the set represented by its subsumer. The subsumee inherits all the structure of its subsumers. The local structure of the subsumee expresses the essential distinctions between it and its subsumers.

The specification of an Object is achieved by creating an ObjectSpec which specifies subsumers for that Object and local structures. The ObjectSpec is then installed in the most specific place and, simultaneously, the most general place possible in the taxonomy. Along with the subsumers specified in the ObjectSpec, additional subsumers may be identified by the classifier. That is, there may be generalizations installed in the taxonomy which were not explicitly specified as subsumers in the ObjectSpec. The classifier may also dissolve an ObjectSpec if it discovers that the ObjectSpec has no properties to distinguish it from an already taxonomized Object.

A Role is described by its subsumers and its local structure which consists of its range and its PrimitiveClasses. Every Role wi'l differentiate its subsumers in at least one of three ways

o Role Conjuntion. If a Role is subsumed by two or more Roles, then it differentiates all of them. The conjunction of two Roles represents fillers which satisfy all relations represented by its parents for a single instance of the domain. In our example, |R|JOB&HOBBY represents the relation between a particular person and an activity that is both his or her hobby and his or her job.

o VRDiff. A Role may differentiate a subsumer by restricting its range to a SubC of its subsumers' range. |R|ATHLETIC-HOBBY is |R|HOBBY with its range restricted to |C|HOBBY-SPORT.

o PrimitiveClass Introduction. A unique PrimitiveClass may be introduced to express how a Role differentiates its subsumer. |R|RELATION subsumes |R|HOME, |R|HOBBY, |R|JOB, and |R|GENDER in ways not accounted for in the taxonomy.

A Concept is described by its subsumers and its local structure which consists of RoleRestrictions of attached Roles, RoleConstraints, and PrimitiveClasses. A Concept must specialize its subsumers in at least one of four ways.

o Concept Conjunction. When a Concept is subsumed by two or more Concepts, it specializes all of them. |C|WOMAN is defined as a SubC of both |C|PERSON and |C|FEMALE-ANIMAL.

o Role Modification. A Concept may specialize its subsumer by creating a new RoleRestriction for an inherited Role. This will restrict the range for the subclass of the domain represented by the Concept in at least one of three ways.

- The VR may be a SubC of the VR of the inherited RoleRestriction. In the first |C|ATHLETE example, shown in Figure 5, |C|ATHLETE is defined as |C|PERSON with the VR of |R|HOBBY restricted to |C|HOBBY-SPORT.
- The Min of the #R may be greater than the Min of the #R in the inherited RoleRestriction. |C|EMPLOYED-PERSON is defined as |C|PERSON with the Min of |R|JOB increased to one.
- The Max of the #R may be less than the Max of the #R in the inherited RoleRestriction.

o RoleConstraint Introduction. The Concept may be the enclosing Concept for a RoleConstraint.

o PrimitiveClass Introduction. A unique PrimitiveClass may be introduced to express how a Concept specializes its subsumer. |C|ANIMAL, |C|PLANT, |C|PERSON, and |C|UNICORN are natural kinds, and so will need a PrimitiveClass.

As mentioned before, there are several descriptors which enhance Object specifications, but do not affect classification: local data, inherited data, disjointness, covering, individual marking, and inverse relations. These may be attatched to taxonomized Objects either before or after classification.

KL-ONE provides a facility to associate keyed data with taxonomized entities. Concepts, Roles, and RoleRestrictions can all have three kinds of attached data. IData is attached to an item and inherited by all its subsumees. Data is local to the item to which it is attached. LocalIData is IData at its most general level of attachment.

The nature of the data is unrestricted; it can be advice, procedures, indications of defaults. Because data can be added to taxonomized entities, a network user can may both access data and add it. Attached data allows a network user to hang information at its most general level af applicability and to distinguish it from information attached at a more specific level.

A DisjointnessClass represents a disjoint set. For Concepts, a DisjointnessClass is a set of Concepts for which there are no common instances. For Roles, a DisjointnessClass is a set of Roles which, for any particular instance in the conjunction of their domains, have no common members in their filler sets.

Each Object in the DisjointnessClass defines a branch of that DisjointnessClass, and all the subsumees of that Object come under that branch. That is, Objects subsumed by different Objects in a DisjointnessClass will also be disjoint. Furthermore, disjointness can be derived for two Concepts when their VRs on the same Role fall under different branches of a DisjointnessClass.

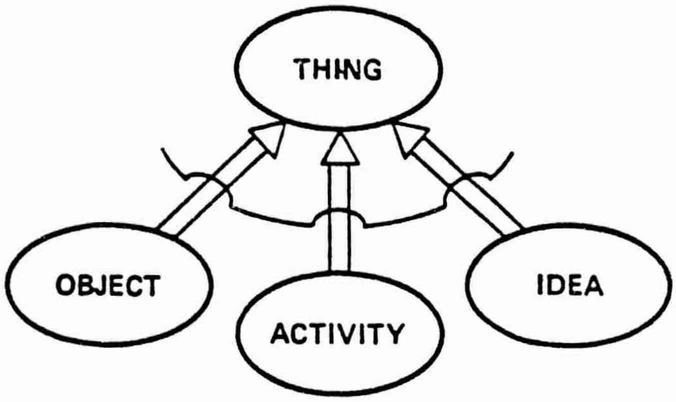


FIG. 12. *A DisjointnessClass with three branches*

Because DisjointnessClasses are independent of classification, there is nothing to prevent an Object from being subsumed by multiple branches of a DisjointnessClass. Such an Object will be marked as incoherent with respect to the appropriate DisjointnessClass.

A Covering is a set of Objects associated with an Object, the covered Object. It expresses that the set represented by the covered Object is exhausted by the sets in the Covering. Every instance of a covered Concept will also be an instance of at least one of the Concepts in the Covering. Similarly, for a particular domain instance, every filler of a covered Role will also be a filler of at least one of the Roles in the Covering for that same domain instance.

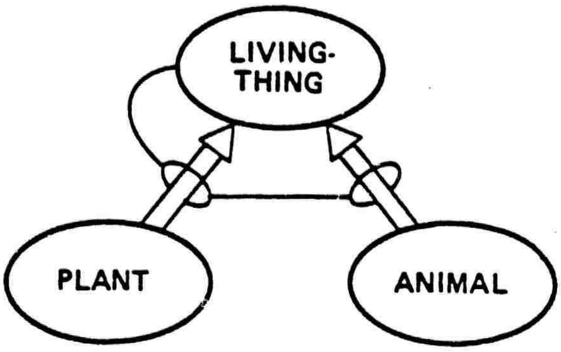


FIG. 13. *A Covering for |C|LIVING-THING*

Because any subset of an exhausted set will also be exhausted, all the subsumees of the covered Object will also be covered. Usually, the Objects in a Covering are subsumed by the most general Object they cover.

Both DisjointnessClasses and Coverings are not used by the KL-ONE classifier, but provide a us'ful reasoning tool for PENNI and any other system accesssing the network. A partition may be expressed with a Covering of disjoint Concepts.

When the relation denoted by a Role has an inverse relation, we can

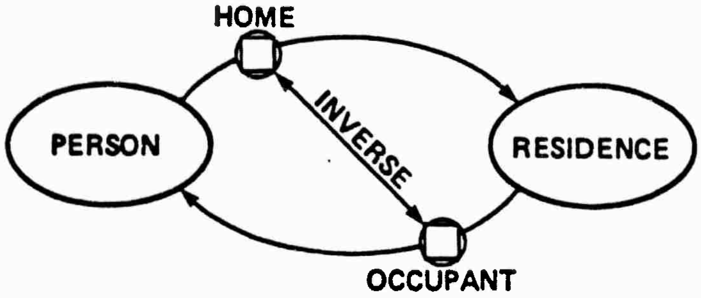


FIG. 14. *A PERSON IS ITS PL HOME 'S OCCUPANT.*
A RESIDENCE IS ITS PL OCCUPANT 'S HOME.

PERSON = OCCUPANT (HOME (PERSON))
& RESIDENCE = HOME (OCCUPANT (RESIDENCE))

express this in KL-ONE by establishing an InverseRole. For any (instance,filler) pair described by a Role, the (filler,instance) pair is described by that Role's InverseRole. This property is inherited by all SubRs of the Role.

In summary, KL-ONE maintains a knowledge base using Concepts to represent classes of things in the world and Roles to represent relations between these classes. Concepts and Roles are interrelated; i.e., Concepts are specified in terms of other Concepts and Roles , and Roles are defined in terms of other Roles and Concepts. Two subsumption taxonomies are maintained, one for Concepts and one for Roles. An Object's place in its taxonomy is defined by creating an ObjectSpec by using a small number of well defined operations to describe it in terms of other Objects. For Concepts, the KL-ONE operations are establishment of SuperC, restriction of Role, and constraint of Role. For Roles, the KL-ONE operations are establishment of SuperR, and restriction of range. When these operations do not establish a sufficient definition of an Object, a PrimitiveClass is introduced. The ObjectSpec is then installed in its taxonomy by the KL-ONE classification operation. In addition, there are several Object components which are not used in classification and so may be established at any time.

3. KL-ONE: SEMANTICS AND CLASSIFICATION[1]

J. Schmolze and D. Israel

3.1 Introduction

Citing Hayes ([6], page 47),

> One can characterize a representational language as one which has ... a semantic theory, by which I mean an account ... of how expressions of the language relate to the individuals or relationships or actions or configurations, etc., comprising the world, or worlds about which the language claims to express knowledge ... Such a semantic theory defines the *meanings* of expressions of the language.

KL-ONE is such a lanquage.

The original designers of KL-ONE were primarily interested in automating the understanding of natural language. They needed a language in which to represent the meanings of sentences (of English). Thus, Brachman et al [1, 2, 7] chose the "real world" as their primary domain and proceeded to design a language in which one could represent knowledge about important classes of real world objects and the relationships between them.

This chapter presents a description of most of the language of KL-ONE. We also specify a semantics for KL-ONE. However, our primary interest is to

[1] A version of this paper has been submitted to the 11th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Salt Lake City, January, 1984.

show some interesting properties of the algorithm for the KL-ONE classifier, which deduces subsumption relationships between the terms of KL-ONE. KRYPTON [4] is the only other representational formalism, in the semantic network style, in which classification plays a central role. For more information plus a description of the system that implements KL-ONE, we refer the reader to [1, 2, 7, 8].

3.2 A Brief Introduction to the KL-ONE Language

KL-ONE lets one define a set of *well formed terms*, which are divided into three groups: *Concepts*, *Rolesets* and *Role-Chains*. Concepts denote properties (i.e., one-place relations) and Rolesets denote two-place relations. Role-Chains are formed by composing either Rolesets or other Role-Chains, and they denote the result of the corresponding relational composition.

Alternatively, one can think of Concepts as denoting sets and both Rolesets and Role-Chains as denoting sets of pairs. A Concept, then, can denote all animals and a Roleset can denote all pairs (a,b) such that b is an offspring of a. By combining these appropriately, one can define a Concept denoting parents, where something is a parent just in case it is an animal and has at least one offspring.

KL-ONE allows for both *primitive* and *defined* terms. The conditions specified for a primitive term are necessary but not sufficient. These terms are used to denote sets for which non-trivial, sufficient conditions for membership cannot be stated, as in sets corresponding to natural kinds. (Of ccurse, KL-ONE does not commit itself to any particular term being primitive.)

The conditions specified for defined terms are both necessary and sufficient, such as the Concept for parents mentioned earlier. Furthermore, the well-formed complex terms are generated by a small set of operators for combining Concepts, Rolesets and Role-Chains. Each of these operators implies a particular meaning for the construct as a function of the meanings of the constituents.

KL-ONE has been implemented as a semantic network in which the terms are represented as nodes and certain relations between terms are represented as links. These relations correspond only to term forming operations. Relations from a particular domain, such as the "offspring" relation, are expressed as terms. This is unlike some other semantic network formalisms that allow domain relations to be expressed as links (see [5, 3, 9]). The most important inter-term relation that KL-ONE maintains is that of subsumption, which in the set-theoretic semantics denotes set inclusion between the sets denoted by the terms. Given the above specifications, the Concept for animals subsumes the Concept for parents, and the KL-ONE system puts a link denoting subsumption between the corresponding nodes.

Thus, a portion of any KL-ONE network is actually a taxonomy based upon subsumption. This is no coincidence - taxonomic reasoning has proven to be extremely useful in the application areas mentioned earlier. It yields a class of inferences that, when done quickly, greatly enhance the performance of such systems. The classification algorithm, mentioned earlier, takes a term and attempts to find all other terms (from a particular, finite network) that either subsume it or that it subsumes. It is a crucial component for our reasoning systems.

3.3 Syntax of the KL-ONE Language

Let K_C denote all of the KL-ONE Concepts, K_R denote all of the Rolesets, and K_{RC} denote all of the Role-Chains. Also, let the Role-Chains include the Rolesets, so K_{RC} includes K_R, and let K be defined as the union of K_C and K_{RC}. A precise definition of K is given below via a set of typed operators (this follows the style of Brachman, et al [4]). Alongside each element and operator is an intuitive description of its meaning (a formal description is in Section 3.5).

There are two distinguished elements of K:

R_{TOP} is in K_R. It denotes the *top* of the Roleset taxonomy, i.e., it subsumes all Rolesets.

C_{TOP} is in K_C. It denotes the *top* of the Concept taxonomy, i.e., it subsumes all Concepts.

KL-ONE allows one to draw from an infinite set of primitive Rolesets and an infinite set of primitive Concepts. The operator RP is defined as a bijection from the natural numbers to the primitive Rolesets such that (RP i) refers to the i-th primitive Roleset. (CP i) is similarly defined for the primitive Concepts.

The definitions of the operators follow. Let i and n be natural numbers. Furthermore, let R, R_1, ..., R_n be elements of K_R, C, C_1, ..., C_n be elements of K_C, and RC, RC_1,..., RC_n be elements of K_{RC}. (The terms "meet", "composition" and "filler" are defined in Section 3.5.)

(RP i) is in K_R anc denotes the i-th primitive Roleset.

(RMeet R_1 ... R_n) is in K_R and denotes the *meet* of Rolesets R_1, ..., R_n.

(RChain RC_1 ... RC_n) is in K_{RC} and denotes a Role-Chain as the *composition* of the Role-Chains RC_1, ..., RC_n.

(CP i) is in K_C and denotes to the i-th primitive Concept.

(CMeet C_1 ... C_n) is in K_C and denotes the meet of Concepts C_1, ..., C_n.

(CRestrict R C) is in K_C and denotes a restriction of the Roleset R to the Value-Description C.

(CMin R n) is in K_C and denotes a Concept with at least n *fillers* of the Roleset R.

(CMax R n) is in K_C and denotes a Concept with at most n fillers of the Roleset R.

(CSubset RC_1 RC_2) is in K_C and denotes a Concept with a subset relation between the fillers of the Role-Chain RC_1 with the fillers of RC_2.

This defines all of K.

While the operators RP and CP refer the primitive Rolesets and Concepts, their relationships to other elements of K are specified via *primitive introductions*.

Let i be a natural number, R be in K_R, and C be in K_C.

(RPrim R i) states that R subsumes the i-th primitive Roleset.

(CPrim C i) states that C subsumes the i-th primitive Concept.

The precise meaning of these operators and restrictions upon their use will be explained in the next two sections.

3.4 Using KL-ONE

We offer some examples of term specifications. Our earlier example for animals and parents is specified with.[1]

```
let ANIMAL = (CP 1);  let Offspring = (RP 1);
let PARENT = (CMeet ANIMAL (CMin Offspring 1))
```

A parent is a animal with at least one offspring, i.e., **PARENT** is subsumed by **ANIMAL** and has at least one filler of the **Offspring** Roleset. If we wanted to state that all mammals were animals and that all people were mammals, we would use primitive introductions:

```
(CPrim ANIMAL 2);  let MAMMAL = (CP 2);
(CPrim MAMMAL 3);  let PERSON = (CP 3);
```

Assuming animals, mammals and people are natural kinds, we denote them by primitive Concepts.

When using KL-ONE, one builds a particular, finite network, i.e., a particular set of Rolesets, Concepts and Role-Chains. A network, called N, is defined to have two parts. N_K is a set of well formed terms of KL-ONE, and is a subset of K. N_P is a set of primitive introductions.

[1] References to terms will be written in bold-face characters. Concepts will be all upper-case; Rolesets will be capitalized.

3.5 A Semantics for KL-ONE

A semantics for a KL-ONE network will be given in a standard first-order language with lambda abstraction (called FOL+). With some network N, we associate a set of predicates, one predicate corresponding to each element of N_K, and a set of sentences in FOL+, one sentence corresponding to each element of N_P.

Before specifying the semantics, we define a notation for expressing number restrictions (i.e., arising from CMin and CMax). Let "[∃n:x][px]" express that there are at least n distinct x's such that each is p-ish.

Our semantic specification consists of two mappings. The first mapping, M, takes each element of N_K into a (possibly complex) predicate, which is denoted in FOL+. The second mapping, AX, takes each primitive introduction into a sentence in FOL+.

M is defined by:

(M R_{TOP}) = lambda xy.x=x&y=y

i.e., the universal two-place predicate

(M (RP i)) = the i-th primitive two-place predicate

which we will write as r^*_i

(M (RMeet R_1 ... R_n)) = lambda xy.(M R_1)xy&...&(M R_n)xy

(M (RChain RC_1 ... RC_n)) =

lambda xy.[∃z_1,...,z_{n-1}]

[(M RC_1)xz_1&(M RC_2)$z_1 z_2$&...&(M RC_{n-1})$z_{n-2} z_{n-1}$&(M RC_n)z_{n-1}y]

(M C_{TOP}) = lambda x.x=x

i.e., the universal one-place predicate

(M (CP i)) = the i-th primitive one-place predicate

which we will write as c^*_i

(M (C[illegible] C_1 ... C_n)) = lambda x.(M C_1)x&...&(M C_n)x

(M (CRestrict R C)) = lambda x.[∀y][(M R)xy->(M C)y]

(M (CMin R n)) = lambda x.[∃n:y][(M R)xy]

(M (CMax R n)) = lambda x.~[∃n+1:y][(M R)xy]

(M (CSubset RC_1 RC_2)) = lambda x.[∀y][(M RC_1)xy->(M RC_2)xy]

A filler for some Roleset R with respect to some Concept C is defined as some y such that "(M C)x&(M R)xy". A filler for a Role-Chain is defined similarly.

AX is defined by:

(AX (RPrim R i)) = "[∀xy][r^*_ixy->(M R)xy]"

(AX (CPrim C i)) = "[∀x][c^*_ix->(M C)x]"

3.6 A Definition of Subsumption

Let N be a network consisting of N_K and N_P, where N_K contains the Concepts C_1 and C_2, and the Rolesets R_1 and R_2. Also, let T_P be the conjunction all sentences associated with N_P via AX. C_1 subsumes C_2 if:

T_P -> [∀x][(M C_2)x->(M C_1)x]

is valid. Intuitively, C_1 subsumes C_2 if every individual that is C_2-ish must also be C_1-ish given the relationships stipulated by the primitive ..troductions.

R_1 subsumes R_2 if:

T_P -> [∀xy][(M R_2)xy->(M R_1)xy]

is valid.

Subsumption is not defined between an element of K_C and an element of K_R. Also, for historical reasons, we have not utilized the relation of subsumption between elements of K_{RC}, as opposed to K_R, although that is an obvious extension we could make.

3.7 The Classifier Algorithm

The classifier algorithm is based upon a function that attempts to determine whether or not two terms stand in a subsumption relation. The function's name is C-SubsumesP (for classifier subsumption) which maps two elements of K into one of *TRUE*, *FALSE* or *INAPPLICABLE*. C-SubsumesP defines a new relation between elements of K which we call c-subsumption.

Using C-SubsumesP, the classifier algorithm takes a newly specified term (call it X) and determines those Concepts from a particular network that X subsumes and those that subsume X. Furthermore, it keeps a record of all subsumptions that it discovers (by adding a link in the network).

Our original hope was that c-subsumption would be identical to subsumption, but our analysis shows that it is not. However, we have shown that C-SubsumesP is sound, i.e., letting X1 and X2 be members of K, then "C-SubsumesP(X1,X2)=*TRUE*" implies that X1 subsumes X2 by the definition in Section 3.6. We have also shown that C-SubsumesP(X1,X2) always terminates.

But C-SubsumesP is not complete, i.e., it is not the case that X1 subsumes X2 implies that "C-SubsumesP(X1,X2)=*TRUE*". There is a class of combinatoric analyses that is not done by C-SubsumesP which must be done in some cases where there are several uses of both CMin and CMax. However, we are hopeful that C-SubsumesP is complete if we eliminate the use of CMax.

Before diving into C-SubsumesP, we first examine an algorithm named Reduce that takes an element of K and reduces it into canonical form. Let X be an element of K.

```
(Reduce X) ==
   (Combine (CSubsetTransClosure (Simplify X)))
```

Simplify "flattens" all terms, i.e., all Concept terms have just one (CMeet ...), or no CMeet at all. Also, all Role-Chains will have only Rolesets as their arguments (not other Role-Chains).

CSubsetTransClosure computes the transitive closure of all uses of CSubset.

Combine puts "like" components together, e.g., (CMeet (CRestrict R C1) (CRestrict R C2)) becomes (CRestrict R (CMeet C1 C2)). It also makes recursive calls to C-SubsumesP in order to eliminate redundant information, e.g., if "C-SubsumesP(C1,C2)=*TRUE*", then (CMeet C1 C2) becomes just C2.

We finally arrive at C-SubsumesP. Various cases for the second argument (the potential subsumee) are specified and each case tests the structure of the first argument (the potential subsumer).

```
(C-SubsumesP $X_1$ $X_2$) ==
  if $X_1$ and $X_2$ are of different types
     then INAPPLICABLE
  else $X_1$ <- (Reduce $X_1$) ; $X_2$ <- (Reduce $X_2$)
   cond
   $X_1=R_{TOP}$: TRUE
   $X_1=C_{TOP}$: TRUE
   $X_2=R_{TOP}$: FALSE
   $X_2=C_{TOP}$: FALSE
   $X_2$=(RP i): $X_1$=(RP i) |
         let (RPrim R' i) be in $N_P$; (C-SubsumesP $X_1$ R')
   $X_2$=(CP i): $X_1$=(CP i) |
         let (CPrim C' i) be in $N_P$; (C-SubsumesP $X_1$ C')
   $X_2$=(CRestrict $R_2$ $C_2$): $X_1$=(CRestrict $R_1$ $C_1$) &
         (C-SubsumesP $R_2$ $R_1$) & (C-SubsumesP $C_1$ $C_2$)
   $X_2$=(CMin $R_2$ $n_2$): $X_1$=(CMin $R_1$ $n_1$) &
         (C-SubsumesP $R_1$ $R_2$) & $n_2 \geq n_1$
   $X_2$=(CMax $R_2$ $n_2$): $X_1$=(CMax $R_1$ $n_1$) &
         (C-SubsumesP $R_2$ $R_1$) & $n_2 \leq n_1$
```

```
$X_2$=(RMeet $R^2_1$ ... $R^2_{n2}$):
    *if* $X_1$=(RMeet $R^1_1$ ... $R^1_{n1}$)
        *then* for each $R^1_j$, (C-SubsumesP $R^1_j$ $X_2$)
    *else* there is some $R^2_i$ st (C-SubsumesP $X_1$ $R^2_i$)
$X_2$=(CMeet $C^2_1$ ... $C^2_{n2}$):
    *if* $X_1$=(CMeet $C^1_1$ ... $C^1_{n1}$)
        *then* for each $C^1_j$, (C-SubsumesP $C^1_j$ $X_2$)
    *else* there is some $C^2_i$ st (C-SubsumesP $X_1$ $C^2_i$)
$X_2$=(RChain $R^2_1$ ... $R^2_n$): {note: n>1, thanks to Simplify}
    $X_1$=(RChain $R^1_1$ ... $R^1_n$) & for $1 \leq i \leq n$, (C-SubsumesP $R^1_i$ $R^2_i$)
$X_2$=(CSubset $RC^2_1$ $RC^2_2$): $X_1$=(CSubset $RC^1_1$ $RC^1_2$) &
        (C-SubsumesP $RC^2_1$ $RC^1_1$) & (C-SubsumesP $RC^1_2$ $RC^2_2$)
```

3.8 Conclusion

We have briefly described the KL-ONE knowledge representation formalism, sketching its syntax and characterizing its semantics. The system is meant to express a certain range of taxonomic or hierarchical relationships among properties and relations, both primitive and defined. Within this framework, the question of an automatic classification scheme arises quite naturally We describe such an algorithm and point toward a proof of its soundness with respect to a defined relation of subsumption between KL-ONE terms[1].

[1] The help and advice of Ron Brachman, Hector Levesque and Krithi Ramamritham is gratefully acknowledged.

REFERENCES

[1] Brachman, R.J. *A Structural Paradigm for Representing Knowledge*. Ph.D. Th., Harvard University, May 1977. Also, BBN Report No.3605, Bolt Beranek and Newman Inc., May 1978

[2] Brachman, R.J., Bobrow, R.J., Cohen, P.R., Klovstad, J.W., Webber, B.L., Woods, W.A. Research in Natural Language Understanding - Annual Report: 1 Sept 78 - 31 Aug 79. BBN Report No. 4274, Bolt Beranek and Newman Inc., Cambridge, MA, August, 1979.

[3] Brachman, R.J. On the Epistemological Status of Semantic Networks. In *Associative Networks - The Representation and Use of Knowledge in Computers*, Findler, Nicholas V., Ed.,Academic Press, New York, 1979.

[4] Brachman, R.J., Fikes, R.E., and Levesque, H.J. KRYPTON: A Functional Approach to Knowledge Representation. techrep 639, Fairchild Research and Development, Artificial Intelligence Laboratory, May, 1983. An extended version to appear in IEEE Computer, Special Issue on Knowledge Representation, September 1983.

[5] Fahlman, S.E.. *NETL: A System for Representing and Using Real-World Knowledge*. MIT Press, Cambridge, Massachusetts, 1979.

[6] Hayes, P.J. The Logic of Frames. In *Frame Conceptions and Text Understanding*, Metzing, Dieter, Ed.,Walter de Gruyter and Co., Berlin, 1979, pp. 46-61.

[7] Schmolze, J.G. and Brachman, R.J. Proceedings of the 1981 KL-ONE Workshop. BBN Report No. 4842, Bolt Beranek and Newman Inc., June, 1982.

[8] Sidner, C.L., Bates, M., Bobrow, R.J., Brachman, R.J., Cohen, P.R., Israel, D., Schmolze, J., Webber, B.L. and Woods, W.A. Research in Knowledge Representation for Natural Language Understanding, Annual Report: 1 September 1980 - 3 August 1981. BBN Report No. 4785, Bolt Beranek and Newman Inc., Cambridge, MA, 1981.

[9] Woods, W.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., Academic Press, New York, 1975, pp. 35-82.

4. KLONEDRAW - A FACILITY FOR AUTOMATICALLY DRAWING PICTURES OF KL-ONE NETWORKS

J. Schmolze

KLONEDRAW is the name of a program that draws pictures of KL-ONE networks. One simply informs KLONEDRAW of which portion of some network to draw and it composes a picture of it using the familiar graphical notation for KL-ONE. Furthermore, this process is completely automatic. Although, we already have several ways of displaying the contents of a network, KLONEDRAW is unique in its role as a pictorial pretty-printer for KL-ONE Concepts.

One uses KLONEDRAW by creating one or more Pictures ("Picture" will be used as a technical term for the following discussion), each of which displays either part, or all, of the current KL-ONE network. The Picture appears to you as an Interlisp window that has a menu alongside. Associated with each Picture is an infinite 2-dimensional plane that we call a blackboard. When you request a Concept to be drawn in some Picture, it is (conceptually) drawn on the blackboard, and the Interlisp window is positioned on the blackboard just over the Concept; thus the drawing of the Concept is visible. The Picture's window is "scroll-able" and can be positioned anywhere on the blackboard. As more and more of the network is drawn, one scrolls the window in order to view different parts of the network. Of course, one can scroll the window semantically as well by requesting that certain Concepts be made visible.

One can have any number of Pictures at any time and they can either have their own, independent blackboards with a drawing of the current network, or they can share a blackboard. The function that creates Pictures, CREATEPICTURE, has an optional argument named PictureForGroup. If PictureForGroup is a Picture, the newly created Picture will share the blackboard of the given one. Of course, the new Picture will have its own window, thereby allowing multiple windows on the same blackboard. Eventually, there will also be a scale argument, letting you create several windows on the same blackboard, each with a different amount of detail. The small version could provide a global view while the normal version gives all of the details. Also, we recently expanded KLONEDRAW to use either a color monitor or the black and white monitor.

4.1 Use of the Mouse

Each part of a KLONEDRAW Picture that corresponds to a KL-ONE object is "mouse sensitive". If you depress the left mouse button, it begins to track your selections and highlights items as you go along, such as Concepts or Roles or SuperC links, etc. This is done by changing the color of the item. As you move away from an item, it is un-highlighted, by going back to their original "color". If you let up the left button over a node or link, you have selected it. Once you select some KL-ONE item, it remains highlighted (but with a slightly different pattern), and the item can be used as an argument to a command from the command menu.

The command menu has several commends which work in the following way. They take zero or more arguments, where the arguments are selected from the corresponding picture. As soon as a command and its required arguments are selected, the command executes, independent of whether you select the command first or the nodes first. Commands are selected in the normal menu way. Among the current list are commands that will re-draw the Picture (which takes zero arguments) or move the drawing of a Concept (which takes a Concept as an argument).

4.2 Functional Interface to KLONEDRAW

The functional interface to KLONEDRAW is so simple that we have included it below:

CREATEPICTURE[TITLE;PICTUREFORGROUP;WINDOW] Creates a KLONEDRAW Picture. A Picture is a structure that includes a blackboard with some portion of the current network drawn on it, a window, and a command menu. TITLE is simply the (optional) title for the window. PICTUREFORGROUP is (optionally) another Picture; if supplied, the Picture being created will have the same blackboard as PICTUREFORGROUP. WINDOW is an optional window to re-use. If no window is supplied, it prompts you for the size of the window.

KLONEDRAW[ENTITYLIST,PICTURE] This is the main function for drawing. ENTITYLIST is either a single entity or a list of entities. An entity is either a KL-ONE object, such as a Concept, or the name of a KL-ONE object. It draws the objects in ENTITYLIST and then scrolls the window so that at least some portion of the items in ENTITYLIST are visible. If the objects in ENTITYLIST are already drawn, the window is simply scrolled so that they are visible. The PICTURE argument determines which Picture is affected.

ERASEPICTURE[PICTURE] Erases the blackboard of PICTURE. Note that if it shares the blackboard with another, only this Picture is affected by this function (i.e., it not longer shares a blackboard).

5. ASSERTIONS IN NIKL

M. Vilain and D. McAllester

5.1 Introduction

The KL-ONE knowledge representation system can be thought of as composed of two subsystems. One part of KL-ONE, the terminological component or Tbox, is responsible for providing a vocabulary of terms with which to describe the world. The Tbox maintains structural and taxonomic relations among these terms. The other part of KL-ONE, the assertional component or Abox, is used to make statements about the world. The Abox records the facts that hold of entities described by terms in the Tbox.

In the past, much of the work done in designing and implementing KL-ONE has focused on the terminological component of the language. The KL-ONE Tbox has developed into a complex and richly expressive system. In contrast, the original KL-ONE system was given a simple assertional mechanism which has not changed substantially since it was first designed. Recent investigations have looked more closely at the KL-ONE Abox and pointed out some important shortcomings in its design [2, 1].

In particular, these investigations voiced dissatisfaction with the limited expressive power of the original KL-ONE Abox. Enhancing this expressive power is the subject of the research described in this article. We are replacing the old KL-ONE assertional mechanism with a considerably more powerful system: at its core is a reasoning engine for the propositional calculus. The new Abox has been named PENN1.[1] PENN1 will serve as the assertional component of NIKL, the new version of KL-ONE that is described in Chapter 2 of this report.

This new Abox significantly changes the nature of the KL-ONE language; in the rest of this article we outline some of these changes. First we review the old KL-ONE assertional mechanism. We then describe the assertional language of PENN1, along with RUP, the propositional reasoning engine on which the new Abox is built. Next we show how the terminological component of NIKL is interfaced to its assertional counterpart. We conclude with a review of how our work has extended the expressive power of KL-ONE.

5.2 The old assertional system

To give perspective on the PENN1 system, we include a brief review of the old KL-ONE assertional mechanism.[2] The old KL-ONE Abox is a simple extension of the terminological taxonomy; hence we begin our description of the Abox with a discussion of the taxonomic component of KL-ONE. Nodes in the taxonomy are descriptive terms; the subsumption relation between concepts is to be taken as relation between such terms. Thus in Figure 1, the node BICYCLIST is to be taken as a description of bicyclists: they have a TRANSPORT-MODE which is BICYCLEs. The superc (subsumption) link between BICYCLIST and PERSON

[1] The name PENN1 is an acronym for the (P)ropositional (EN)gine for (N1)KL.

[2] The description of the old Abox given here is necessarily cursory. More details can be found in [8].

states that anything which can be described as a BICYCLISſ can be described as a PERSON — the structure in Figure 1 can be read as saying "All bicyclists are persons".

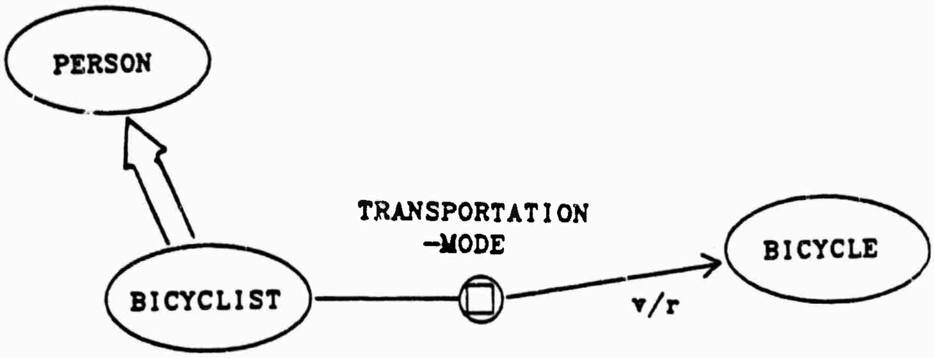


FIG. 1. A KL-ONE NETWORK

Absent from these descriptive readings of taxonomic terms is any claim that the terms describe entities which actually exist. The network in Figure 1 makes no statement about the actual existence of bicyclists or persons. To provide a mechanism for notating existence, the constructs of nexus nodes and description wires were added to the KL-ONE language.

At the heart of the old Abox is the construct of nexus nodes. A nexus node in a KL-ONE network stands for an entity in the world; it denotes that entity. Unlike the descriptive nodes of the concept taxonomy, the nexus nodes are taken assertionally: a nexus in a network stands for an individual which is asserted to exist. Nexus nodes are connected into the network by means of description wires. These wires have the following reading: if a nexus N is connected by a description wire to a concept C, then the entity denoted by N is described by C. Consider the first network in Figure 2. It has the following interpretation: the nexus John denotes some individual in the world, and this individual can be described as a BICYCLIST. We can read this structure as asserting "John is a bicyclist". As the second network in Figure 2 suggests, a nexus can be described by more than one concept by running several wires out of the nexus. The second network can be read "John is a bicyclist and a red-haired person".

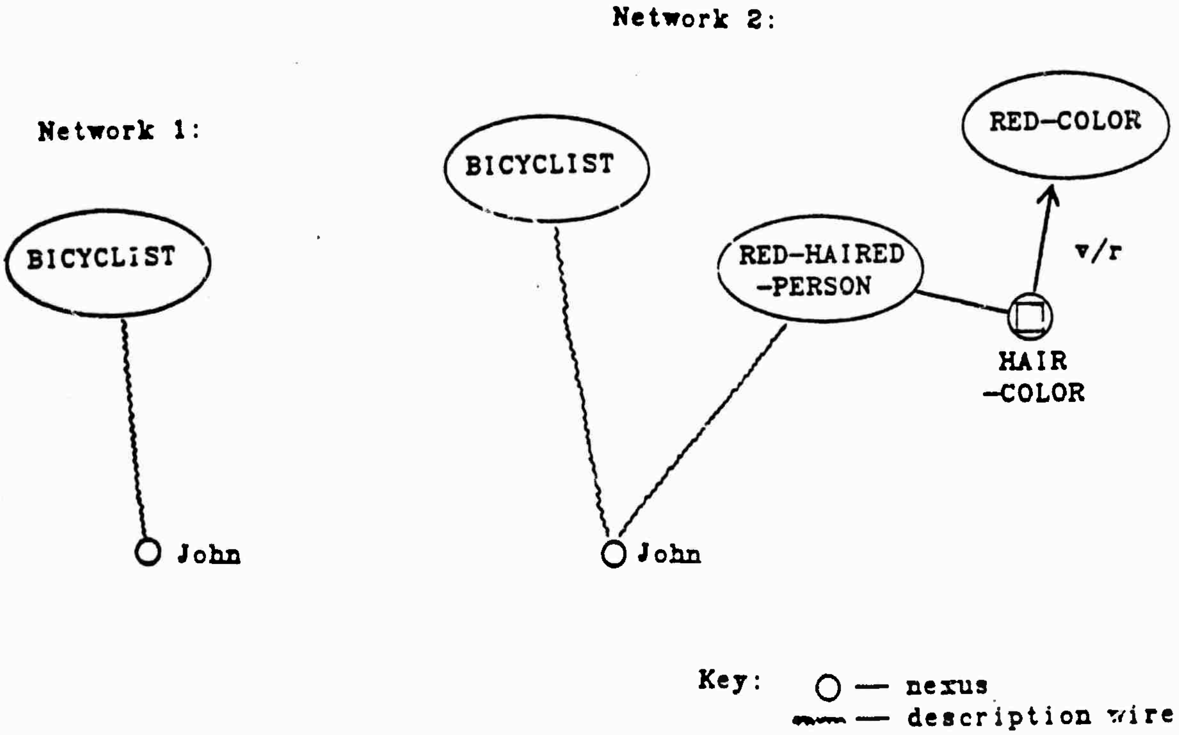


FIG. 2. A NETWORK WITH NEXUS NODES

5.3 Problems end Proposals

The original KL-ONE Abox was a very simple mechanism, and its simplicity made it a straightforward and clean extension of the terminological part of KL-ONE. However, this very simplicity also posed great restrictions on the kinds of assertions that could be formed in KL-ONE. The following are some of the more salient shortcomings of the old Abox.

- In the old assertional language it isn't possible to make "weak", or not fully determinate, statements about individuals. One can not assert disjunctive propositions (e.g., "John is either a bicyclist or a motorcyclist"). Nor can one assert negated propositions (e.g., "John is not a motorcyclist"). This is a weakness of the description wire scheme: there are no special kinds of description wires that encode disjunction or negation.

- The nexus nodes of the old Abox must be interpreted as denoting *distinct* individuals. It isn't possible to equate two nexus nodes — that is, one can't assert that the two nodes actually denote the *same* individual. For instance, say we construct a nexus (called VENUS) that denotes the second planet orbiting our sun, and construct another node (called MORNINGSTAR) denoting the star that appears on the horizon every morning. If we later learn that these two celestial bodies are one and the same individual, we are at a loss to express this. Because the nexus nodes must denote different individuals, we can not equate VENUS and MORNINGSTAR.

- The old Abox doesn't provide a way to assert propositions that have the status of inference rules. By this we mean that the language can not express implications (e.g., "if John is a bicyclist then he is not a motorcyclist"). Nor can it be used to express a wide range of quantified statements (e.g., "every bicyclist is not a motorcyclist").

- Finally, the old Abox doesn't provide any mechanism for making inferences automatically from statements in the assertional language.

Each of these shortcomings of the old KL-ONE Abox constitutes a major restriction on the expressive power of the KL-ONE assertional language. In fact, taken together these shortcomings define some appealing desiderata for the revised Abox of NIKL.

These desiderata resemble "a prescription ... for a language like that of First Order Predicate Logic" [2]. This view very much embodies the philosophy behind PENNI, the assertional component of NIKL. We have replaced the old assertional language of nexus nodes and description wires with a language based on a fragment of the predicate calculus.

As was said at the start of this paper, statements in the assertional language are sentences formed out of terms in the taxonomic language.[1] Because the A-language is based on predicate logic, we must agree to a formal (logical) reading of the T-language. Elsewhere in this report [9], we sketch a formal semantics for the taxonomic language. This account defines the way taxonomic terms are used in the A-language. Briefly, we proceed as follows.

For each concept node in the taxonomy, we identify a corresponding 1-place predicate which has the same name as the concept. For example, to the PERSON and BICYCLIST concepts of Figure 1, correspond respectively the A-language predicates named "PERSON" and "BICYCLIST" — as in (PERSON John) or (BICYCLIST John). Similarly, each role in the NIKL role lattice has associated a 2-place predicate bearing its name. Thus to the TRANSPORTATION-MODE role in Figure 1 corresponds the A-language predicate named "TRANSPORTATION-MODE". Finally, constant symbols denote individuals in the world (e.g., the constant symbol "John" in (BICYCLIST John) denotes some individual person). Constant symbols correspond to the nexus nodes of the old assertional language.

[1] We will also use the terms "A-language" (or "PENNI language") and "T-language" (or "NIKL" language) to refer to the assertional and taxonomic languages respectively.

Constant symbols in the A-language denote individuals in the world. To assert that a term in the T-language, i.e. a NIKL concept, describes an individual, one applies to the individual the A-language predicate corresponding to the T-language term. This statement is then an assertion in the A-language. To illustrate this, the following table contains some examples of PENNI language assertions. The assertions in the table cover many of the examples given earlier during the discussion of the old nexus node mechanism. The table also contains examples of A-language statements that could not have been expressed using the old Abox. Note that the examples refer to concept terms taken from the taxonomy in Figure 3.

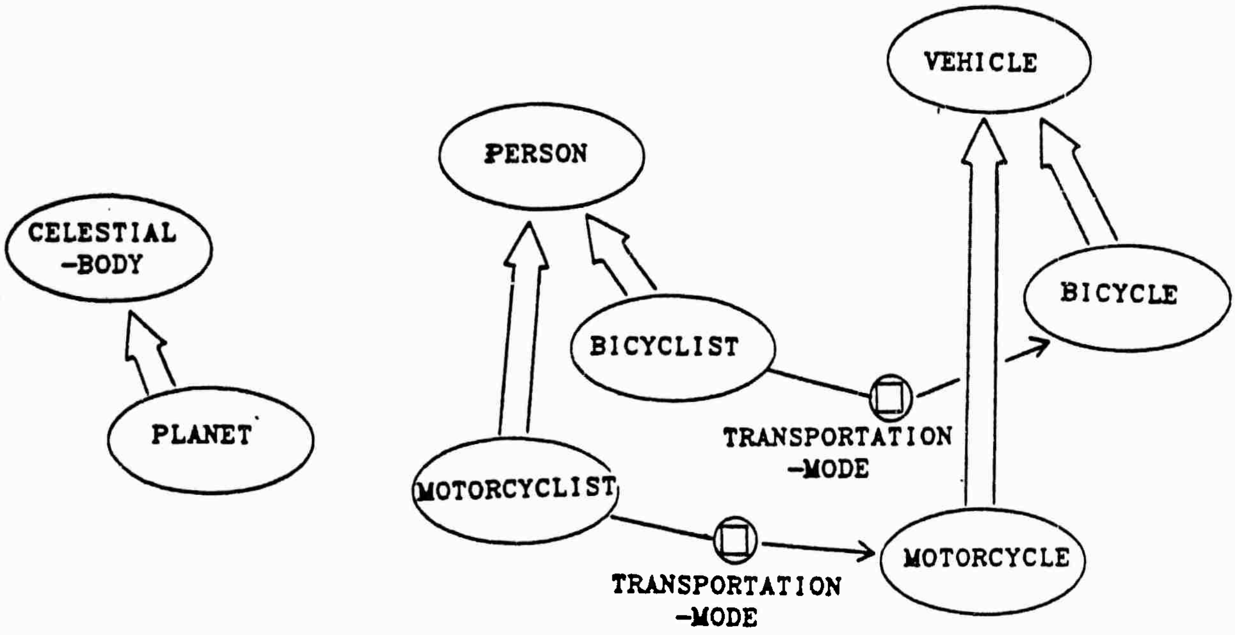


FIG. 3. THE TAXONOMY USED IN TABLE 1

"John is a bicyclist"	(BICYCLIST John)
"TrustyRusty is a bicycle"	(BICYCLE TrustyRusty)
"John's transportation is TrustyRusty"	(TRANSPORTATION-MODE John TrustyRusty)
"John is a bicyclist and a red-haired person"	(AND (BICYCLIST John) (RED-HAIRED-PERSON John))
"John is either a bicyclist or a motorcyclist"	(OR (BICYCLIST John) (MOTORCYCLIST John))
"John isn't a motorcyclist"	(NOT (MOTORCYCLIST John))
"if John is a bicyclist then he isn't a motorcyclist"	(=> (BICYCLIST John) (NOT (MOTORCYCLIST John)))
"Venus is a planet"	(PLANET Venus)
"The morning star is a celestial body"	(CELESTIALBODY MorningStar)
"Venus and the morning star are one and the same"	(= Venus MorningStar)

TABLE 1. A-LANGUAGE EXAMPLES

These examples are suggestive of the scope of the PENNI language. More precisely, the assertional language is exactly defined by these four conditions:

1. (P a) is an assertion in the A-language, where P is the predicate corresponding to some NIKL concept (We will use the terms "NIKL predicate" or "NIKL-pred" to refer to this kind of predicate.), and a is an individual term.

2. (R a b) is an assertion in the A-language, where R is a relation corresponding to a NIKL role (We will use the terms "NIKL relation" or "NIKL-rel" to refer to this kind of relation.), and both a and b are individual terms.

3. (= a b) is an assertion in the A-language, where both a and b are individual terms.

4. Boolean combinations of assertions in the A-language are themselves assertions in the A-language. By these Boolean combinations we mean statements of the form (NOT P), (OR P Q), (AND P Q), (=> P Q), or (<=> P Q), where P and Q are both A-language assertions.

5.4 RUP, the system underlying PENNI

In the preceding section we described the assertional language of the PENNI system. In this section we will discuss some of the functionality of PENNI. In particular, we will look at RUP[1], a powerful reasoning system on top of which PENNI was built.

RUP is a system that was developed at MIT by David McAllester. The system consists of a set of reasoning utilities that are designed to underly knowledge representation systems. Within PENNI, RUP is used to maintain a database of A-language assertions and to perform inferences on these assertions. In the pages that follow, we will describe some of the features of RUP that are relevant to PENNI. We will gloss over many (significant) details about RUP; the interested reader is referred to [6] and [5].

5.4.1 The RUP truth maintenance system

At the center of RUP is the RUP truth maintenance system, or TMS. The TMS contains a database of formulae of the propositional calculus. Associated with each proposition is a truth setting which indicates whether the proposition is held to be true, false, or unknown (a proposition with a truth setting of unknown is a proposition which the system doesn't know to be true or false). These three truth settings are the only ones that can be assigned to a proposition.

The user can enter propositions into the TMS incrementally. As each

[1] The name RUP is an acronym for the (R)easoning (U)tility (P)ackage.

proposition is entered, the user can assign it a truth setting, thereby asserting the proposition to be true or false.[1] When the user asserts a proposition, the TMS invokes an inference engine to derive the consequences of the assertion. This antecedent or premise driven inference engine deduces a subset of the consequences entailed by the addition of the new proposition. It is significant that the consequences which are actually deduced are only a subset of those which are entailed. Trying to deduce all of the consequences could lead to an exponential effort. RUP restricts the inferences that it tries to make and thereby achieves considerable efficiency.[2]

To illustrate the use of the TMS, consider the following example. Say we add to the TMS the proposition:

```
(=> (BICYCLIST John) (NOT (MOTORCYCLIST John)))
```

This results in the creation of four TMS data structures (called TMS nodes). One of these TMS nodes correspond to the asserted implication, and the other three correspond to its component sub-expressions. This situation is depicted in Figure 4.

Note that in Figure 4 the only proposition that has been given a determinate truth setting is the one that was asserted. The premise- driven

[1] The user can also leave the proposition with a truth setting of unknown; in this case the system will try to deduce a truth setting for the proposition from other propositions in the database.

[2] However, RUP does provide a consequent or goal driven reasoning mechanism that can capture deductions that were not made by the antecedent engine. For details see [6] and [5].

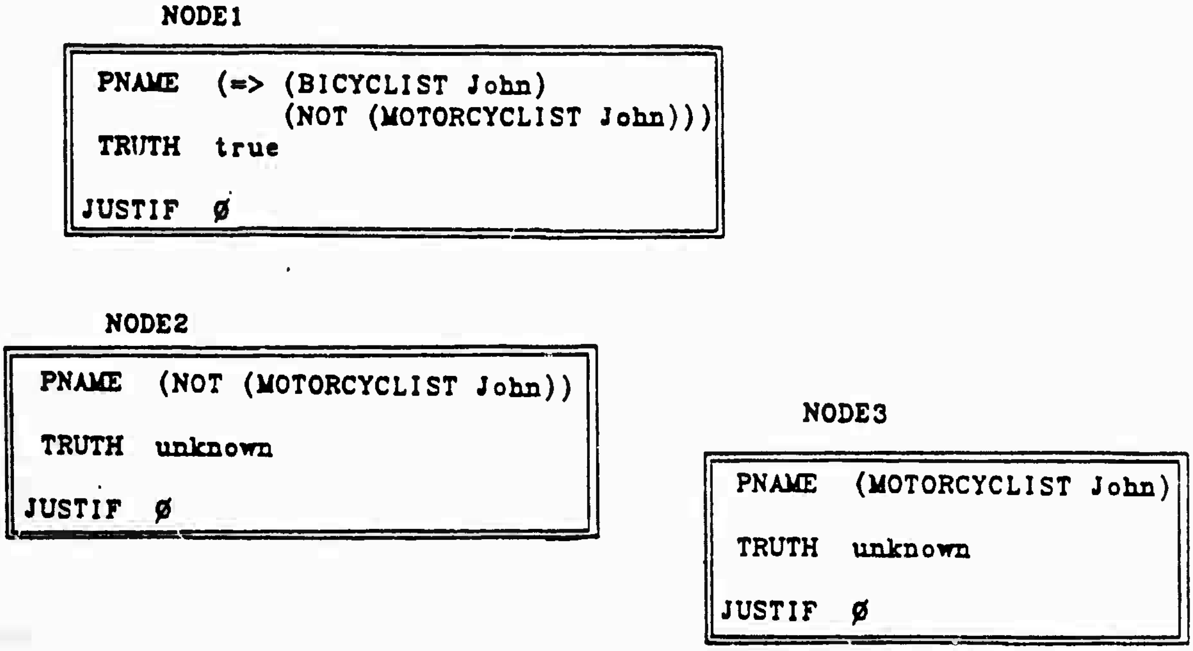

NODE1
PNAME (=> (BICYCLIST John)
(NOT (MOTORCYCLIST John)))
TRUTH true
JUSTIF ø
NODE2
PNAME (NOT (MOTORCYCLIST John))
TRUTH unknown
JUSTIF ø
NODE3
PNAME (MOTORCYCLIST John)
TRUTH unknown
JUSTIF ø

FIG. 4. AN ASSERTED IMPLICATION

engine was (justifiably!) unable to deduce truth settings for the sub-expressions of the asserted proposition.

To continue the example, say we now assert the proposition:

(BICYCLIST John)

The TMS invokes its inference engine which can now make several deductions. On the basis of the proposition we just added and the one asserted earlier the system deduces that

(NOT (MOTORCYCLIST John))

must be true. On the basis of this new inference, the system can further deduce that

(MOTORCYCLIST John)

must be false. The database ends up as in Figure 5.

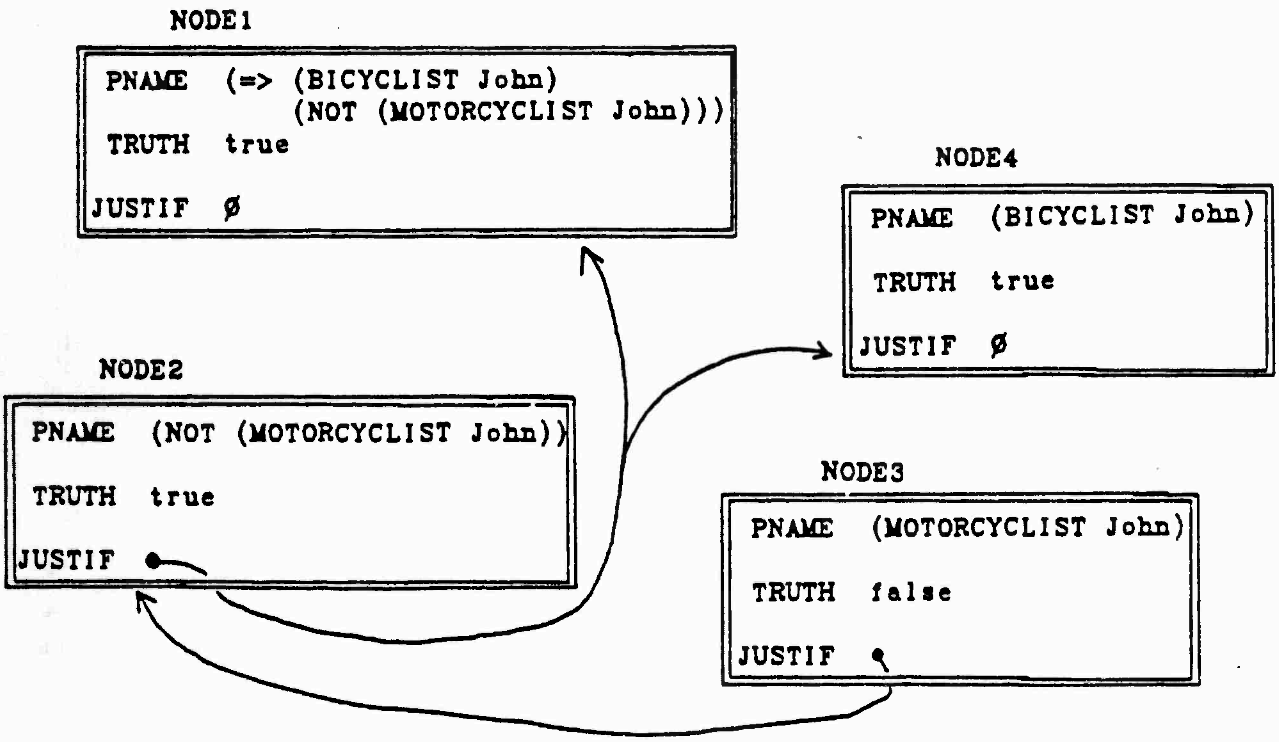


FIG. 5. AN INFERENCE IN THE TMS

With each deduction made by the system, a record is kept of which propositions in the database were used in making the deduction. This is accomplished through the justification field of TMS nodes. Given a proposition *P* in the database, the justification field of its TMS node contains pointers to a set of propositions which together entail *P*. If *P* was asserted by the user, then the justification field of *P* is set tc NIL. Alternatively, if *P* was deduced by the system, the field points to those propositions used by the system in deducing *P*. This is illustrated by Figure

5. Among others, the figure depicts the justifications recorded by the system when running the example described above.

The justifications that RUP records for its deductions are a very important aspect of the system. RUP uses the justifications it records to provide the user with a number of sophisticated features. In particular:

- RUP can use its recorded justifications to explain to the user how it arrived at a particular conclusion. It does this by searching through the justification pointers for the exact set of user assertions that underly the conclusion. This set is returned to the user.
- RUP uses the justifications to perform efficient incremental retraction.
- RUP also uses the justification pointers in its backtracker. The backtracker implements dependency directed backtracking, an extremely efficient backtracking technique.

We will not describe these features in further detail here. The interested reader is referred to [6] and [5] for more information.

5.4.2 The RUP equality system

In the preceding paragraphs we described features of the RUP truth maintenance system. We now turn our attention to another component of RUP, the equality system.

The equality system is responsible for maintaining a congruence relation between terms in the TMS database. The equality system groups congruent terms into congruence classes. It uses these congruence classes for performing substitution of equals for equals in propositions stored in the database.

The equality system is activated by assertions in the database of form

(= TERM1 TERM2).

When the user asserts such a proposition, the equality system enforces the assertion by placing the two equated terms in the same congruence class. This is done according to the following scheme. When a term is first entered in the database, a new (singleton) congruence class is created and the term is made the sole member of the class. When two terms are equated, the equality system fetches their respective congruence classes and merges the two classes, producing a single composite class. In this operation it doesn't matter whether the classes being merged are singleton classes or contain more than one member. This scheme ensures that any two congruent terms are always in the same congruence class. Consequently, to test whether any two terms are congruent, the equality system only has to check whether their congruence classes are the same.

To illustrate the equality system, consider the following example. Say we assert these propositions:

```
(CELESTIAL-BODY MorningStar)
(CELESTIAL-BODY EveningStar)
(PLANET Venus)
```

Figure 6 portrays the resulting state of the RUP system. In the left half of the figure are the TMS nodes corresponding to the three propositions. In the right half are three "Venn diagram bubbles"; they correspond to the singleton congruence classes for the three terms MorningStar, EveningStar, and Venus.

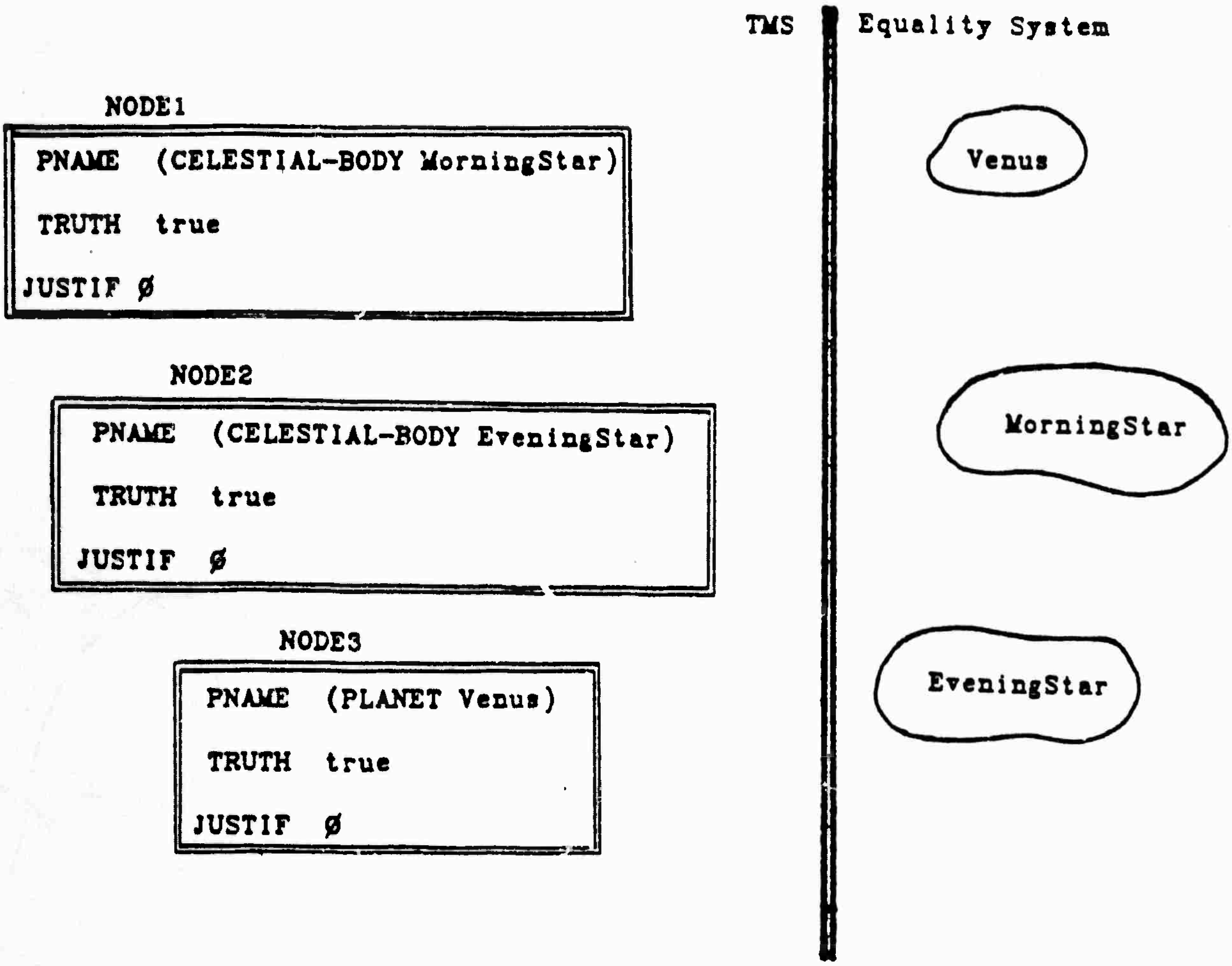


FIG. 6. TMS NODES AND CONGRUENCE CLASSES

Say we now add these assertions to the database:

```
(= Venus MorningStar)
(= Venus EveningStar)
```

These additions will cause a sequence of events to occur. The first assertion causes the equality system to fetch the congruence classes for the terms Venus and MorningStar; it merges these two classes. A similar operation for Venus and EveningStar is brought about by the second assertion, leaving all three terms Venus, MorningStar, and EveningStar in the same congruence class. This is depicted by Figure 7.

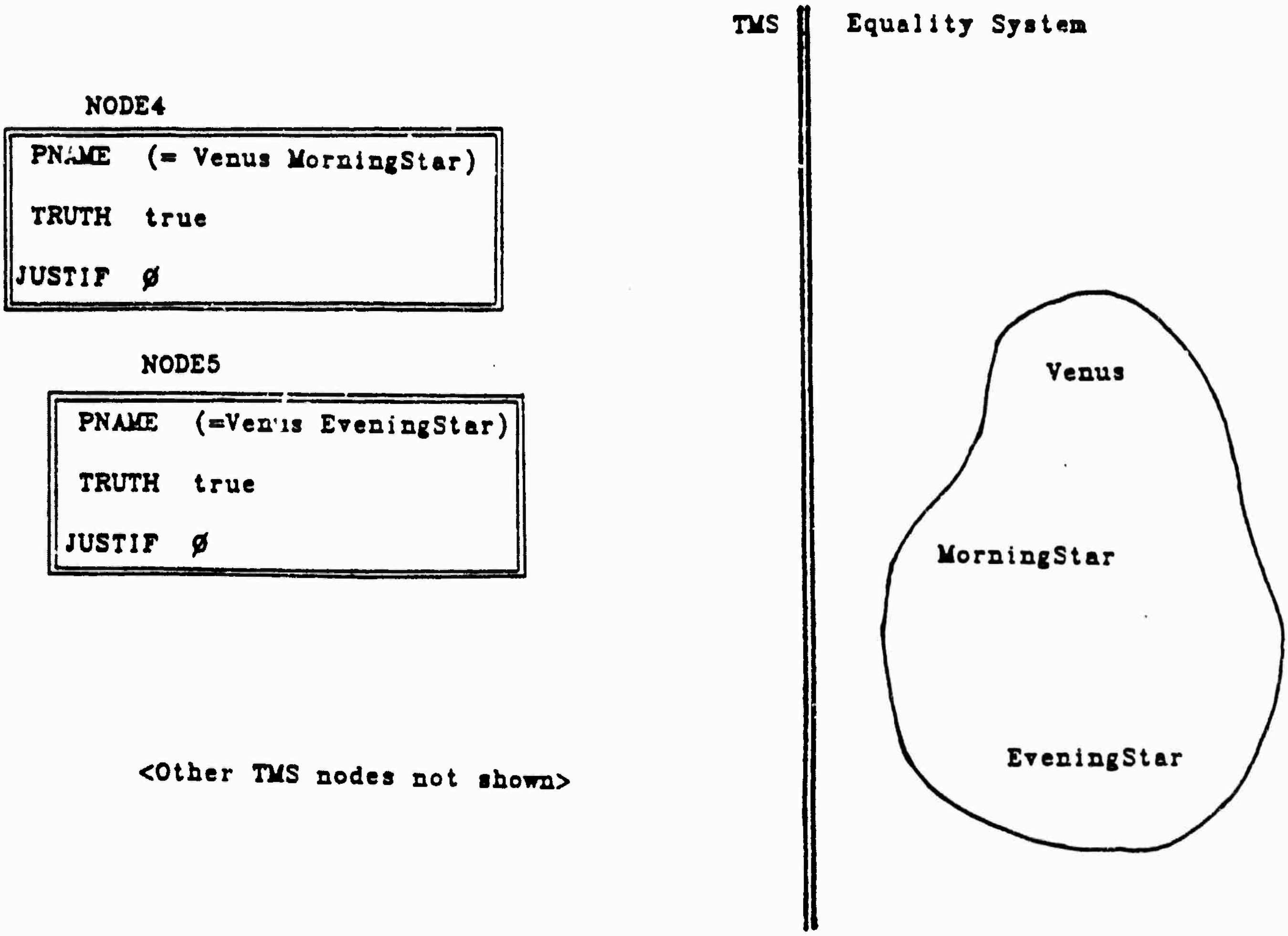


FIG. 7. EQUALITY ASSERTIONS

Figure 7 illustrates an important characteristic of the equality system. Notice that even though the terms "EveningStar" and "MorningStar" are in the same equality system congruence class, there is no TMS node in the database corresponding to the proposition:

(= MorningStar EveningStar)

The equality system didn't on its own enter the proposition in the database, even though the proposition was entailed by the two earlier equality assertions. This is by design; it prsvents the database from being overloaded with equality propositions.

If we actually want to test whether EveningStar and MorningStar are congruent, it suffices for us to enter the proposition

(= MorningStar EveningStar)

into the database with a truth setting of unknown. Entering the proposition in the database invokes the equality system which then tries to derive a truth setting for the proposition. This derivation is done by simply comparing the congruence classes for MorningStar and EveningStar. Since the classes are one and the same, the equality system gives the proposition a truth setting of true, and leaves the database as in Figure 8.[1]

Our discussion of the equality system so far has centered on equality assertions and the congruence relation between terms. The equality system also performs substitution of equals for equals. Consider again the example of Venus and the Evening Star. In particular, consider the assertions:

(PLANET Venus)

[1] Note that the equality system installed justification pointers for the proposition (= MorningStar EveningStar). The details of how this is done are beyond the scope of this paper. Once again, the reader is referred to [6] and [5].

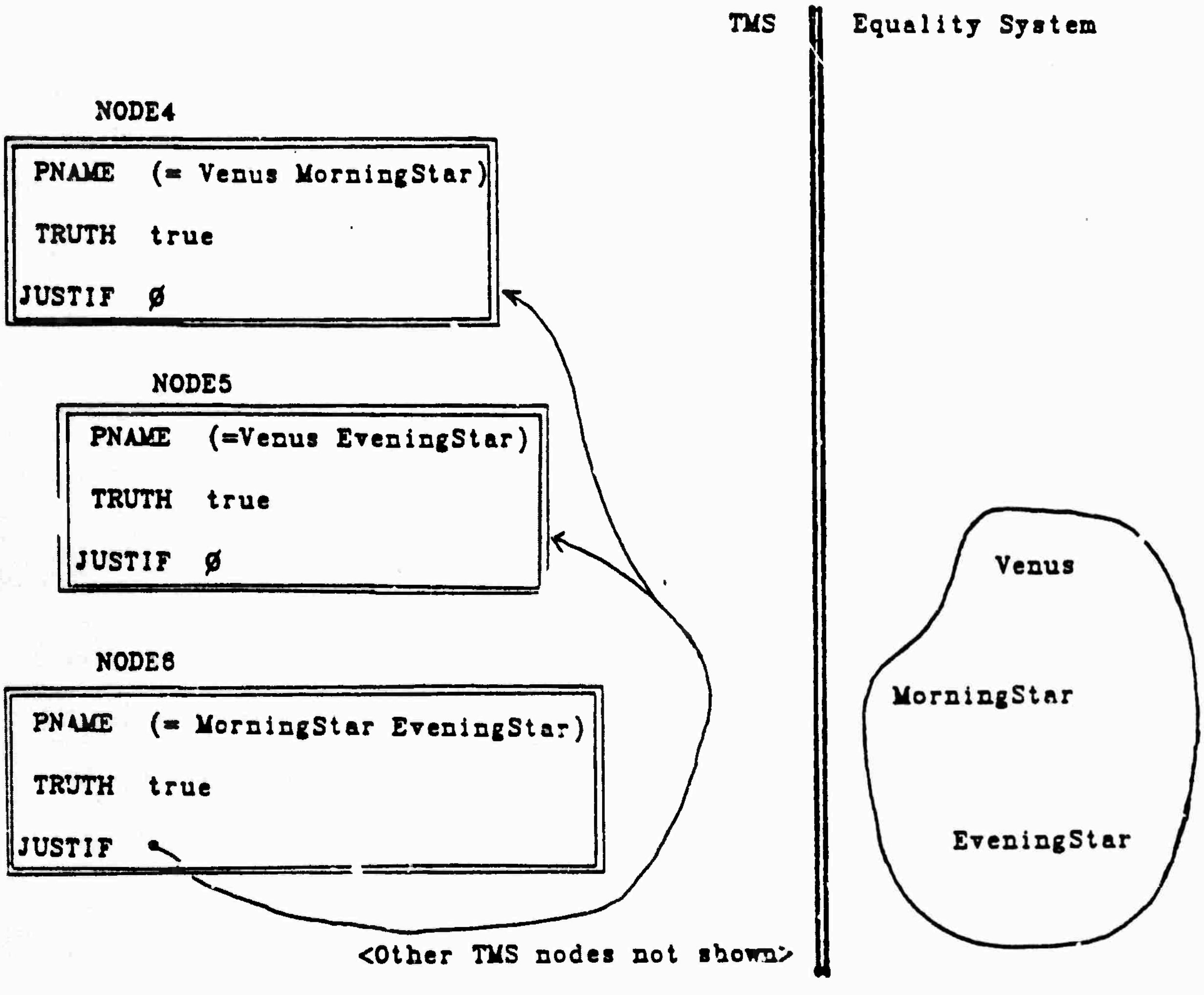

TMS
Equality System
NODE4
PNAME (= Venus MorningStar)
TRUTH true
JUSTIF ∅
NODE5
PNAME (=Venus EveningStar)
TRUTH true
JUSTIF ∅
NODE6
PNAME (= MorningStar EveningStar)
TRUTH true
JUSTIF
<Other TMS nodes not shown>
Venus
MorningStar
EveningStar

FIG. 8. AN INFERENCE BASED ON EQUALITIES

(= Venus EveningStar)

Together, they entail another proposition: (PLANET EveningStar). This proposition follows from substitution of equals for equals. However, the equality system does not add this proposition to the database when Venus and EveningStar are equated. This is for the same reason that it doesn't assert all the equality propositions that follow from user-asserted equalities: the equality system tries to prevent overloading the database with unnecessary propositions.

If we want to query whether (PLANET EveningStar) is true, we proceed once again by adding the proposition to the database with a truth setting of unknown. This activates the equality system which then attempts to derive a determinate truth setting for the proposition. To do so, the equality system recognizes that EveningStar is in the same congruence class as Venus. From this it follows that

(PLANET EveningStar)

is logically equivalent to

(PLANET Venus).

The two should have the same truth setting. Since the second of these is assigned a truth setting of true, the equality system infers that the first one should be true as well. Figure 9 shows the resulting state of the system.

5.4.3 The RUP noticer compiler

The final feature of RUP which we will describe here is the noticer compiler. The noticer compiler allows the user to write demons similar to those in the PLANNER system. These demons (which we call noticers) are LISP functions which are invoked when certain events occur within the RUP database. Characteristic of the events that will trigger noticers are the addition of a proposition to the database, a change in the truth setting of a proposition, and others.

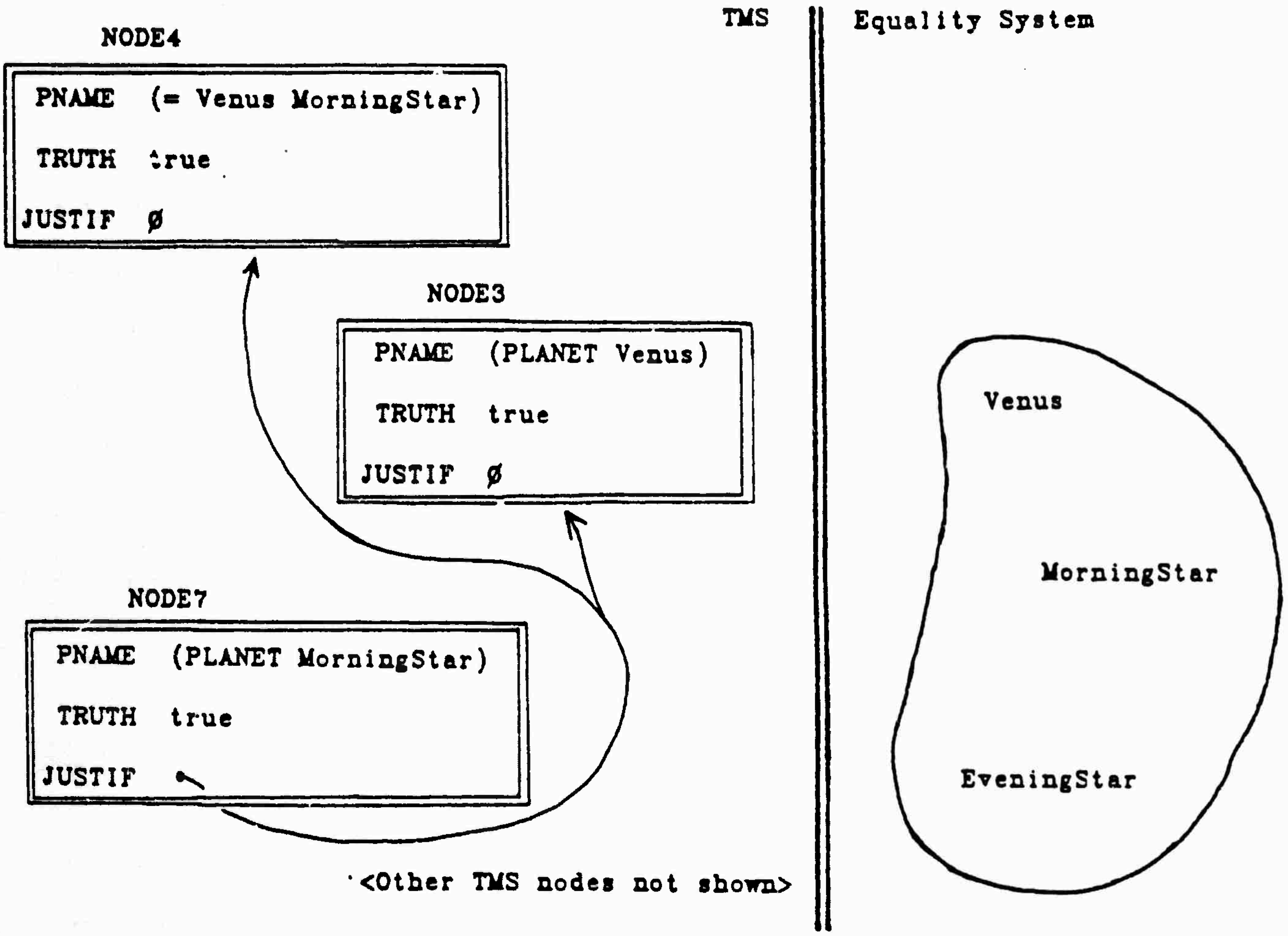


FIG. 9. SUBSTITUTION OF EQUALS FOR EQUALS

When a noticer is defined by the user, it must be given two primary components: a trigger condition and a body. The trigger condition indicates in which situations the noticer should be invoked. The body of the noticer is the LISP code that gets executed when the noticer is activated.

The trigger of a noticer has two components: a pattern that matches propositions in the database and an event marker which specifies a database event (such as assertion or truth change). When an event specified by the event marker of a trigger occurs to a proposition that matches the trigger's pattern, the trigger is activated and the noticer associated with the trigger is invoked. The RUP noticer compiler provides a language which facilitates defining the trigger conditions of noticers. The compiler compiles these trigger conditions into internal RUP tests and actions that encode them.

When a noticer is activated it is run just like any other LISP function.[1] In practice, the noticer will often perform some specific computation and then (if needed) make changes to the RUP database to reflect the results of this computation. The language provided by the noticer compiler facilitates making these changes. The compiler compiles statements that manipulate the database into optimized invocations of internal RUP functions.

We will not try in this paper to give a complete description of the features of the noticer compiler. Instead we will simply illustrate the use of the noticer system by giving an extended example of a noticer definition. This is a very simple noticer that recognizes predications of the form (PLANET ?x). It then asserts that if ?x (or rather, that term which matches ?x) is a planet, then it must orbit the sun. The noticer is particularly simple in that it doesn't perform any computation before manipulating the database. For the sake of clarity, the following definition strays somewhat from the syntax which is actually recognized by the noticer compiler.

[1] This is a simplification. In actual fact, when a noticer is triggered it gets placed in a queue of activated noticers. Later on, the queue will get emptied and the noticer will be run.

```
(DEFNOTICER  NoticePlanets
    (TRIGGER (PLANET ?x) INTERN)
    (BODY    (RUPAssert  (=> (PLANET ?x)
                             (ORBITS ?x Sol)))
    ))
```

DEFNOTICER is a function that defines RUP noticers. It is passed three arguments. The first is the name of the noticer: noticers (just as functions) are given a name — in this case the name of the noticer is "NoticePlanets". The second and third arguments to DEFNOTICER are the trigger and body of the noticer being defined.

In this example, the trigger condition of the noticer specifies the pattern (PLANET ?x) and the event INTERN. The pattern matches any one-place predication whose predicate name is PLANET. The INTERN event marker indicates that the noticer should be invoked when propositions that match the trigger pattern are INTERNed, i.e. entered into the database. Note that the ?x term in the trigger pattern is taken as a free variable. It gets bound to the argument of any one-place PLANET predication that activates the noticer.

For example, say we add the following proposition to the database:

(PLANET Venus)

This proposition matches the trigger pattern of the NoticePlanets noticer; the ?x free variable in the trigger pattern gets bound to the term Venus.

In the trigger pattern of a noticer, any atomic term that begins with the letter "?" is treated by RUP as a free variable that will get bound at the time the noticer is activated. This binding is maintained while the noticer is executed and is made available to the LISP code in the body of the noticer. To continue our example, consider the body of the NoticePlanets noticer. This body contains a call to the function RUPAssert, the basic function for asserting propositions in RUP. The argument to RUPAssert is a proposition that mentions the variable ?x:

```
    (=> (PLANET ?x) (ORBITS ?x Sol))
```

When the noticer got activated by our entering (PLANET Venus) into the database, the ?x variable got bound to the term Venus. Hence, when RUPAssert actually gets called in the body of NoticePlanets, its argument is the fully instantiated proposition:

```
    (=> (PLANET Venus) (ORBITS Venus Sol))
```

RUPAssert adds this proposition to the database with a truth setting of true. Since we had just asserted (PLANET Venus) to be true, the new addition causes (ORBITS Venus Sol) to be deduced true as well.

This concludes our discussion of the noticer compiler, and along with that our discussion of RUP.

5.5 The PENNI System

We will now resume our description of the PENNI system and the assertional language it provides. So far, this paper has described two aspects of our work on PENNI. We have talked about the assertional language provided by PENNI, and we have discussed RUP, the propositional reasoner underlying PENNI. In the pages to come, we will fit these two views of our work together. We will first describe briefly how RUP is used in PENNI. We will then consider how PENNI fits within the NIKL effort as a whole. In particular, we will show how the inference mechanism for the A-language is interfaced to the terminological component of NIKL.

The wav PENNI uses RUP is actually very straightforward. As we alluded to earlier, PENNI uses RUP to maintain a database of A-language assertions. Within RUP, these assertions are not treated any differently from other propositions in the RUP database. Just as with any other proposition, A-language assertions can be incrementally asserted or retracted; they may serve to justify or be justified by other propositions; they may be involved in noticer invocations; and so on. Thus, PENNI uses the full features of RUP to implement its database of A-language assertions. Hence, at some level, much of the functionality that PENNI provides within NIKL is simply that functionality which is provided by RUP. However, the A-language database is only one aspect of the features provided by the PENNI system. The remainder of PENNI's functionality stems from PENNI's position as part of a larger knowledge representation system — NIKL.

Within NIKL, the Abox and the Tbox are distinct, but closely coupled subsystems. They are distinct in that each can be used independently: the former to perform assertional inferences, and the other to perform terminological inferences. However, the two subsystems are intended to be used together, as a connected whole. The connection between PENNI and the Tbox lies in the fact that PENNI recognizes when it can use terminological inferences from the Tbox to supplement assertional inferences. This interconnection of the terminological and assertional com,onents of NIKL is a crucial feature of the system. Most of the remainder of this paper will be spent analyzing the Abox-Tbox connection and looking at how providing this connection affects the use of NIKL.

5.5.1 The PENNI-NIKL interface

To appreciate the coupling between the asserticnal and terminological components of the system, we must return for a moment to our original description of the Abox-Tbox distinction. The terminological component of NIKL, the Tbox, is responsible for providing a vocabulary of structured terms with which to describe the world. The domain over which the Tbox reasons is this vocabulary of terms: in some sense the Tbox is <u>about</u> terms. On the other hand, the assertional component of NIKL, the Abox, uses terms from the Tbox vocabulary to build models of the world. In particular, it uses the Tbox terms to say things about individuals in the world. In the same sense that the Tbox is <u>about</u> terms, the Abox is <u>about</u> individuals.

In NIKL, it is individuals (in the Abox) that serve as the locus of the interface between the assertional and terminological components of the system. For each individual x mentioned in the assertional database, PENNI keeps a record of all the assertions made of x. From this, PENNI isolates that particular set of assertions that consists of NIKL-predications and NIKL-relations made of x that are assigned a truth setting of true.[1]

[1] NIKL-preds and NIKL-rels that are assigned truth settings of false or unknown are treated differently. We will not go into the details of this here.

Each of these NIKL-preds and NIKL-rels corresponds to a term in the Tbox taxonomy — a concept or a role. PENNI fetches these corresponding Tbox terms and combines them to produce a new composite term. This composition is a straightforward taxonomic operation and proceeds in the following way:

1. Suppose PENNI is constructing the composite Tbox term for the individual *x*. Further, say the following NIKL-predications and NIKL-relations are true of *x*:

   ```
   (C1 x)
   ...
   (Cn x)
   (R1 x y1)
   ...
   (Rm x ym)
   ```

2. First, PENNI constructs a new concept in the Tbox taxonomy which will be the composite term for *x*. (Note: we will also call this composite term the composite concept for *x* or the composite description of *x*.)

3. For each NIKL-pred *(Ci x)* asserted of *x*, PENNI retrieves the corresponding concept in the taxonomy, *Ci*. This concept is made a subsumer of the composite description of *x*.

4. For each NIKL-rel *(Rj x yj)* involving *x*, PENNI retrieves the corresponding NIKL role, *Rj*, and attaches information about this role to the composite concept for *x*. There are technical intricacies as to how this is done — among these: the composite concept for each of the *yj* must be computed before the role information can be attached.

   The result of this first set of operations is depicted in Figure 10.

5. PENNI next enters the composite term it has just constructed into the Tbox taxonomy; this is done by invoking the Tbox classifier. The classifier will ensure that the new term is entered into its most specific (appropriate) location in the taxonomy.[1]

[1] For details about how the classifier performs this operation, see [4] and [9].

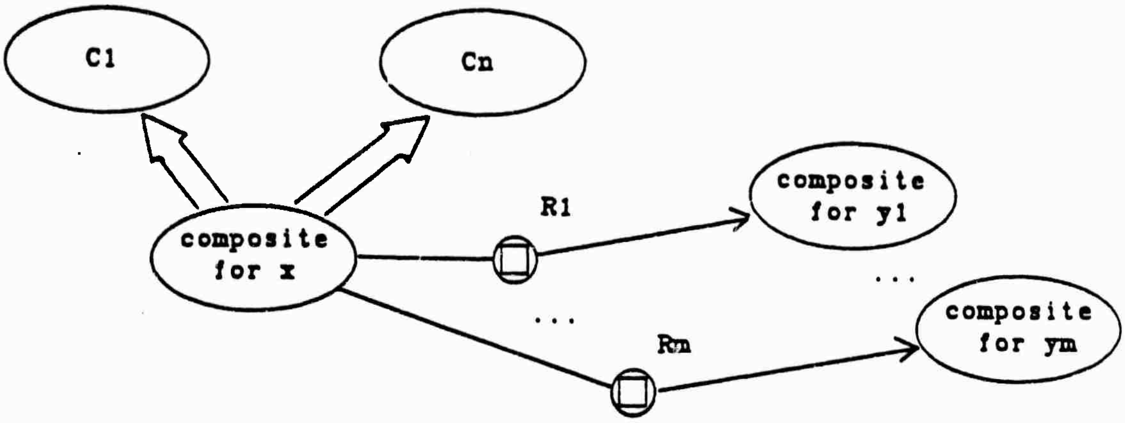


FIG. 10. THE COMPOSITE CONCEPT FOR *X*

The process we have just described results in the creation of a term in the Tbox taxonomy. Starting from Abox assertions about an individual *x*, PENNI produces a concept in the taxonomy which we take to describe *x*. By nature of the classifier, this new concept is in fact the most specific concept which can be deduced (from the taxonomy) to describe *x*. We call this concept the MSG of *x*; this stands for the most specific generic concept that describes *x*.

PENNI records the association between an individual and its MSG by adding a NIKL-predication to the assertional database. The predicate that is added is simply the NIKL-pred that corresponds to the MSG of the individual. For example, if the MSG that was constructed for an individual *x* is the concept BICYCLIST, PENNI adds to the database the assertion (BICYCLIST x).

Given the association between individuals and their MSGs, the Abox can now invoke the Tbox to make terminological inferences that have bearing on

Abox individuals. Several kinds of inferences can be made in this way. Some inferences follow directly from the very action of using the classifier to enter an individual's MSG into the taxonomy. Others are implicit given the position in the taxonomy of an individual's MSG. These latter inferences are not made until the need for them arises. In the paragraphs below we illustrate some of these inferences by means of a few extended examples.

5.5.2 An example involving classification

As we mentioned above, some terminological inferences that get reflected in the Abox are immediate consequences of classifying the MSGs of individuals. As an example of this, consider the network in Figure 11. This network contains three concepts: BICYCLIST, RED-HAIRED-PERSON, and RED-HAIRED-BICYCLIST. The third concept is a subconcept of the first and second concepts. What is more, the RED-HAIRED-BICYCLIST concept is taken to be a defined concept. It is defined to be that concept which describes exactly those individuals that are described by both the BICYCLIST and RED-HAIRED-PERSON concepts.

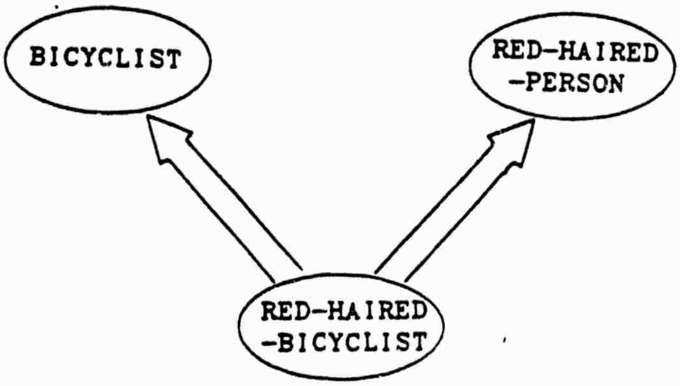


FIG. 11. A NETWORK WITH A DEFINED CONCEPT

Say some user program now asserts the following propositions in the PENNI database:

(BICYCLIST John)
(RED-HAIRED-PERSON John)

PENNI collects these two assertions about the individual John and uses them to construct the MSG for John. This MSG is depicted in Figure 12. It has two superconcepts: BICYCLIST and RED-HAIRED-PERSON. Note that this MSG is once again a defined concept; in fact, it is defined to be exactly the same concept as RED-HAIRED-BICYCLIST. When the MSG of John is given to the classifier, the classifier recognizes that the two concepts are defined identically. The classifier "merges" the MSG concept created by PENNI and the RED-HAIRED-BICYCLIST concept. This results in the individual John's MSG now being the concept RED-HAIRED-BICYCLIST, as shown in Figure 13.

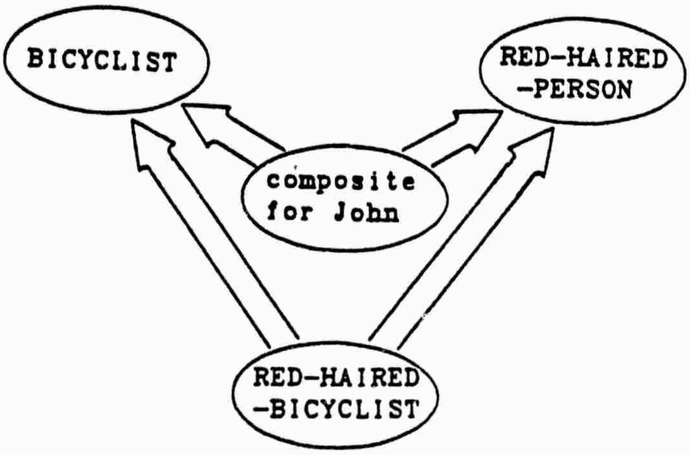


FIG. 12. THE COMPOSITE CONCEPT FOR *JOHN*

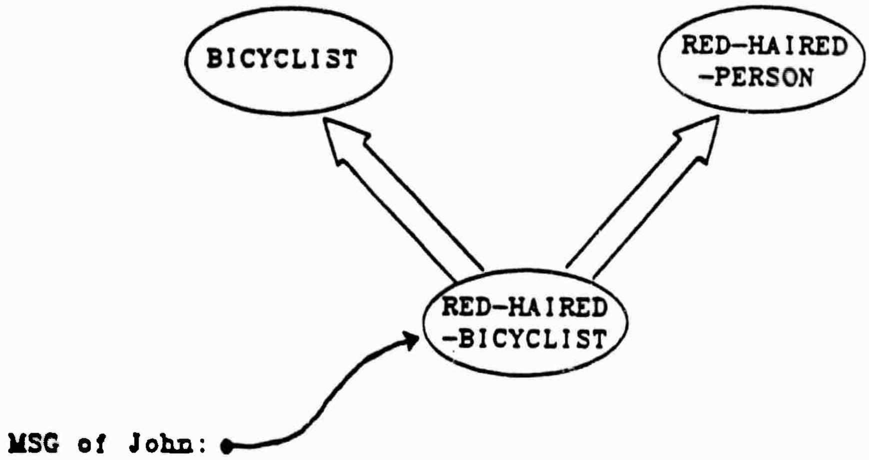


FIG. 13. THE MSG FOR *JOHN*

After this classification step, PENNI associates John and its MSG by adding the following proposition to the A-language database:

(RED-HAIRED-BICYCLIST John)

Adding this proposition to the database has several significant consequences. In particular, the predicate RED-HAIRED-BICYCLIST may have a particular significance to the user program that made the original assertions (BICYCLIST John) and (RED-HAIRED-PERSON John). Given that (RED-HAIRED-BICYCLIST John) has now been asserted in the database, this user program may now be able to make new inferences.

One way in which these inferences could be made is by using noticers. For example, the user program we've been mentioning could have defined some noticer with the trigger pattern

(TRIGGER (RED-HAIRED-BICYCLIST ?x) INTERN)

This noticer would have been activated when PENNI associated John and its MSG.

5.5.3 An example involving subsumption

The preceding example showed an instance of a terminological deduction that PENNI added automatically to its assertional database. This inference was added to the database as soon as PENNI had an opportunity to do so — that is, immediately after invoking the classifier on an individual's MSG. Not all of the terminological inferences that PENNI makes use of are performed in this way. Many inferences are left implicit. One class of these implicit inferences are those inferences which are based on subsumption.

Consider the network in Figure 14. The network consists of the two concepts BICYCLIST and PERSON. BICYCLIST is subsumed by PERSON, and we read this subsumption relation (as usual) as a universally quantified implication.

$\forall x$ (BICYCLIST x) => (PERSON x)

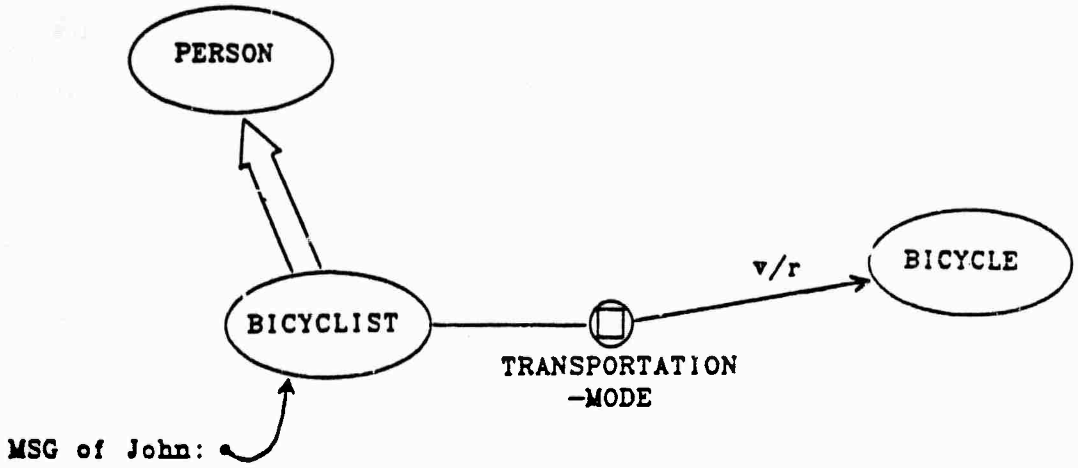


FIG. 14. A NETWORK WITH THE MSG FOR *JOHN*

Say some user program now asserts

(BICYCLIST John).

The MSG of John is trivially computed to be the concept BICYCLIST; this is shown in Figure 14. Say the user program now wishes to query the truth of the proposition

(PERSON John).

To do this, the program simply enters the proposition into the database with a truth setting of unknown. When the proposition is interned, PENNI recognizes it to be a NIKL-predication of the individual John. PENNI then tried to derive a truth setting for the proposition by using the Tbox. It does this by first fetching the Tbox term corresponding to the predication — this term is simply the concept PERSON. PENNI then asks the Tbox whether this concept (i.e. PERSON) subsumes the MSG of the individual John (i.e. BICYCLIST). This is indeed the case, and given this PENNI can now assign a truth setting of true to the proposition (PERSON John).

Note that PENNI does not automatically infer the proposition (PERSON John) after calculating the MSG for John. The user program had to intern the proposition for the deduction to be made. This is by design: by following this scheme PENNI avoids adding unnecessary predications to the assertional database. (The alternative to this scheme would be to add to the database a NIKL-predication corresponding to every concept in the taxonomy which subsumes an individual's MSG — all the way up to THING. This is impractical and grossly inefficient.)

Inferences based on subsumption should to be thought of as implicit. They are never made explicitly until they are needed; when they are needed they are made automatically.

5.5.4 An example involving roles

In this example we will look at an inference that PENNI makes using role information. Consider once again the network in Figure 14. In particular, consider the TRANSPORTATION-MODE role attached to BICYCLIST. This role is value-restricted to the concept BICYCLIST. We read the restriction information as follows:

$$\forall x,y\ [(\mathrm{BICYCLIST}\ x)\ \&\ (\mathrm{TRANSPORTATION\text{-}MODE}\ x\ y)] \Rightarrow (\mathrm{BICYCLE}\ y)$$

Say some user program now asserts these predications:

```
(BICYCLIST John)
(TRANSPORTATION-MODE John TrustyRusty)
```

After the first assertion, the MSG for the individual John is trivially computed to be the concept BICYCLIST. When the second proposition is added, PENNI recognizes it to be an assertion of a NIKL-relation between the individuals John and TrustyRusty. PENNI then fetches the role (from the Tbox role lattice) corresponding to the relation, and looks up the range restriction of the role. This range restriction is computed using the domain restriction provided by the MSG for John. As we just saw, this range restriction is simply the concept BICYCLE.

PENNI reads this restriction as the same universally quantified statement we gave above. This allows PENNI to infer the proposition

(BICYCLE TrustyRusty)

which is then added to the propositional database. This addition has all the usual consequences of asserting a NIKL-predication: the MSG for the individual TrustyRusty will be updated (or created), and further Abox inferences may occur.

5.5.5 What do these examples have in common?

Each of the examples we have just seen shows the Tbox being used to supplement the Abox. In each example, the NIKL taxonomy was used to perform terminological deductions that were then reflected in the assertional database. In some cases, this enabled further inference to proceed in the Abox.

None of these terminological deductions could have been performed easily using the Abox alone. To capture the full scope of the terminological reasoning done by the Tbox would require adding to the Abox many rules of inference. This would prove to be a substantial burden. In fact, no general purpose inference engine could be expected to do this reasoning efficiently. By separating assertional and terminological reasoning, NIKL is able to provide efficient inference mechanisms for both.

5.6 Conclusion

We started our investigation of the new NIKL Abox with a list of desiderata. How does the work we have just described measure up to these wishes? We feel it measures up well. In particular:

- The new assertional language supports "weak" statements. By providing propositional calculus as the basis for the A-language we allow for not fully determinate statements.

- The new Abox supports reasoning about equality. Abox individuals can be equated, thereby asserting them to denote the same entity in the world.

- The new Abox supports inference rules. By judicious use of noticers, the user can write rules that encode some quantified reasoning.

- The new Abox provides a powerful inference engine. This inference engine is RUP, whose array of inference utilities makes it one of the most sophisticated and versatile reasoning systems currently in existence.

We feel that our efforts towards building the assertional component of NIKL have met with some success. However, the efforts we have described here are just the beginning of our investigation. The development of PENNI is an ongoing process, and much work still remains to be done. This work isn't limited to our research alone. Indeed, the idea of separating a general knowledge representation system into distinct, but interacting, subsystems, is gaining serious acceptance in the field of AI. Other knowledge representation systems now exist that, like NIKL, distinguish different kinds of knowledge and provide separate, but coupled, inference engines for each [3, 7]. We expect that this approach will yield many contributions to knowledge representation. NIKL and PENNI are among the first of these.

REFERENCES

[1] Bobrow, Robert J., David J. Israel, and James G. Schmolze. The NIKL Working Papers. unpublished, Bolt Beranek and Newman Inc.

[2] Brachman, Ronald J. and Hector J. Levesque. Assertions in KL-ONE. In *Proceedings of the 1981 KL-ONE Workshop*, Schmolze, James G. and Ronald J. Brachman, Eds., BBN Report No. 4842, 1982, pp. 8-17.

[3] Brachman, Ronald J., Richard E. Fikes, and Hector J. Levesque. "KRYPTON: A Functional Approach to Knowledge Representation." *IEEE Computer 16* (October 1983). Also available as Fairchild Technical Report No. 639 or FLAIR Technical Report No. 16.

[4] Lipkis, Thomas. A KL-ONE Classifier. In *Proceedings of the 1981 KL-ONE Workshop*, Schmolze, James G. and Ronald J. Brachman, Eds., BBN Report No. 4842, 1982, pp. 128-145.

[5] McAllester, David A. An Outlook On Truth Maintenance. Massachusetts Institute Technology, Artificial Intelligence Laboratory, August, 1980. AI Memo No. 551

[6] McAllester, David A. Reasoning Utility Package User's Manual. Massachusetts Institute Technology, Artificial Intelligence Laboratory, April, 1982. AI Memo No. 667

[7] Rich, Charles. Knowledge Representation Languages and Predicate Calculus: How to Have Your Cake and Eat It Too. Proceedings of the AAAI-82, American Association for Artificial Intelligence, 1982, pp. 193-196.

[8] Schmolze, James G., and Ronald J. Brachman. Summary of the KL-ONE Language. In *Proceedings of the 1981 KL-ONE Workshop*, Schmolze, James G., and Ronald J. Brachman, Eds., BBN Report No. 4842, 1982, pp. 233-260.

[9] Schmolze, James G. and Israel David J. KL-ONE: Semantics and Classification. In *Research in Knowledge Representation for Natural Language Understanding, Annual Report September 1982 - August 1983*, Bolt Beranek and Newman Inc, Report No. 5421, 1983.

6. BELIEF AND KNOWLEDGE IN ARTIFICIAL INTELLIGENCE[1]

A. Haas

6.1 Representation and Search

Artificial Intelligence programs must have common-sense knowledge. This includes knowledge about beliefs and knowledge. A program must be able to understand that Bill believes Mary's phone number is 5766, or that John knows the name of every person in the department. If a program is supposed to understand these facts, it should be able to make the right inferences from them. If Bill knows that Mary's phone number is 5766, he knows what Mary's phone number is. If a program thinks that Bill knows that Mary's number is 5766, the program should be able to infer that Bill knows what Mary's number is. If we have a knowledge representation that can represent facts about beliefs and knowledge, and an adequate set of inference rules, we have taken the first step in building a program that can reason about beliefs and knowledge. The next step is to devise a search strategy: an algorithm that decides which inference rules to apply to which expressions to solve a problem.

This paper is about the first step. It proposes a representation and

[1] A version of this paper has been published as BBN Technical Report No. 5368 "The Syntactic Theory of Belief and Knowledge" and, under this same title, has been submitted for publication in a collection, edited by Prof. Jerome Feldman, in the series Advances in Artificial Intelligence.

inference rules for reasoning about belief and knowledge. Section 6.2 presents examples of sound and unsound inferences about belief and knowledge. The problem is to allow all the sound inferences and rule out the unsound ones. The best treatment to date is Moore's [13], and I discuss his successes and failures. Section 6.4 presents the syntactic theory of belief and shows how to formalize it. Section 6.5 is the core of the paper: a series of examples of representation and inference in the formal system. These examples describe the processes that create, store and use beliefs and knowledge. Perception, introspection, memory, inference and planning are all considered.

Recent work has made great improvement in AI theories of belief and knowledge, but they still have serious problems. For example, Moore's theory predicts that agents always know every logical consequence of their knowledge. My theory tries to solve the problems by formalizing familiar ideas from computer science. For example, it says that sentences stored in an agent's memory represent his beliefs. It makes three main improvements in the match between theory and common sense. First, it does not predict that agents always infer everything that follows from their knowledge. It considers the agent's goals and his limited inference ability before predicting that he will make an inference, and it says that inference takes time. Second, it gives a better account of what you must know about an object in order to know what that object is. It says that you know what an object is if you know enough about it to carry out your intended actions. Finally, it gives a better account of when you need knowledge to perform an action. It simply formalizes the obvious: robots perform actions by sending commands to effectors, and to act they must find out which commands will produce the desired actions.

These improvements have practical importance. A planner will not get far if (following Moore's theory) it thinks that there is no point in planning to do inferences, since they all happen instantly and automatically. Nor will an interactive program do well if it thinks that a large mathematical expression is a good answer to a user's question because it is a standard name.

These improvements are all made in the same way: by using familiar ideas from computer science. If an agent uses sentences to represent his beliefs, and applies inference rules to them, there is no reason to expect that he will believe all consequences of his beliefs. If an agent acts by sending commands to his effectors, then of course he must find out which commands will produce the desired actions. Konolige took this line, but he only went halfway - he returned to Moore's theory in his treatment of knowing what and knowing how. As a result, the problems of Moore's theory reappear in Konolige's theory.

There is also an important gain in the technique for reasoning about another agent's inferences. The idea of building a data base to represent another agent's beliefs has always appealed to AI workers. But it had a serious problem: there was no way to represent that John knows what Mary's phone number without putting Mary's phone number in the data base. The use of new constants to stand for unknown terms solves this problem.

6.2 Some Inferences About Belief and Knowledge

Let us consider some examples that show why reasoning about beliefs is hard. For one thing, the familiar rule of substitution of equals does not apply when one of the equals appears inside the scope of the verb "believe". For example, the following inference is not valid.

John believes that Mary's phone number ıs 444-1212.
Bill's phone number is Mary's phone number.

John believes that Bill's phone number is 444-1212.

It is easy enough to forbid the substitution of equals when one of the equals appears inside the scope of "believe", but this is not very satisfying. One would like an explanation of why substitution of equals does not apply.

The following inferences are valid:

John knows that snow is white.

John believes that snow is white.

John knows that snow is white.

Snow is white.

That is, all knowledgc is true belief. On the other hand, not all true beliefs are knowledge. Suppose somebody predicts that a horse will win a race when the odds are 30 to 1 against it. Sure enough, the horse wins. We might ask "How did he know the horse would win?". It would make sense to answer "He didn't know, it was just a lucky guess." That is, a true belief might not count as knowledge if there is no good reason for the belief. I will not consider this problem further. Suffice it to say that all knowledge is true belief.

The following inference is valid.

John knows that Mary's phone number is 444-1212.

John knows what Mary's phone number is.

But this one is not necessarily valid:

John knows that Mary's phone number is Bill's phone number.

John knows what Mary's phone number is.

This raises the question: when does John's knowing that X is N entail that John knows what X is? The noun phrases "444-1212" and "Bill's phone number" both denote Mary's phone number, but knowing that Mary's number is Bill's number does not count as knowing what Mary's phone number is. In some sense the phrase "Bill's phone number" does not contain enough information, but it's hard to clarify this.

Context helps to decide what knowledge about X counts as knowing what X is. Suppose that you and John are staying at a hotel in a strange city, and you go out for a walk. After a while John asks "Do you know where we are?" You realize that you're completely lost, and answer "No." Seeing a telephone you decide to call Mary and ask for directions. She answers and says "Do you know where John is? I need to talk to him right away." You answer "Yes, he's right here" and hand him the phone. When John asked if you knew where he was you said no; a moment later you answered yes to the same question.

If you had answered John's question with "Yes; we're right here", he would not have been amused. John wanted information that would help him to get back to the hotel. Mary wanted information that would help her to get in touch with John, and for that purpose "right here" was a useful description of John's location.

One clue to the problem of "knowing what" comes from the problem of "knowing how". The following inference is correct:

John knows that Mary's number is 444-1212.
John knows how to dial a telephone.

John knows how to dial Mary's number.

This one is not:

John knows that Mary's number is Bill's number.
John knows how to dial a telephone.

John knows how to dial Mary's number.

We saw that if you have the name "444-1212" for Mary's number you know what her number is, but not if you only have the name "Bill's number". Similarly, if you have the name "444-1212" for Mary's number you know how to dial the number, but not if you only have the name "Bill's number". It is tempting to connect these two facts. In any case a theory of belief and knowledge must say something about what knowledge is needed to perform actions. So the theory of belief and knowledge is connected to the theory of planning.

The problem of "knowing what" is closely related to the so-called de re statements about belief. Suppose you see John in a restaurant with a woman you don't know, and you think "That must be John's wife". Later you find she was his sister. You might say "I thought John's sister was his wife." The following inference is valid, at least in some contexts:

I thought John's sister was an accountant.

I believed the statement "John's sister is an accountant".

But the following is surely not valid in this context.

I thought John's sister was his wife.

I believed the statement "John's sister is his wife".

In this case the example seems to mean about the same as "I saw John's sister and thought she was his wife". The speaker uses the description "John's sister" to identify the woman he took for John's wife. Such statements are called de re reports of belief or knowledge.

Truth is a crucial property of beliefs. Our theory must explain inferences like this:

John believes that gold is an element.
Everything that John believes is true.

Gold is an element.

If we know that someone's beliefs are true, we can infer things about the objects those beliefs refer to. We can also reason in the other direction: if an object has certain properties, then certain beliefs about it are true.

Coal is black.
John believes that coal is black.

John believes something true.

Common sense says that we think about objects outside our heads, and that our beliefs about them can be right or wrong.

People use their beliefs to infer new beliefs. What they infer depends on what problems they want to solve and how hard they think. For example, the following inference is very plausible.

John knows that Mary's number is 5766.
John knows that Mary's number is Bill's number.
John is trying to figure out what Bill's number is.

John will infer that Bill's number is 5766.

On the other hand, a math teacher had better not accept the following:

The students believe the Axiom of Choice.
The Axiom of Choice entails that every set can be well-ordered.

The students will infer that every set can be well-ordered.

A theory of belief ought to distinguish hard inferences from easy ones, and it ought to say that what people infer from their beliefs depends on what they try to infer.

People know about their own beliefs. They can easily answer questions like "Do you know what Mary's phone number is?". Yet we don't want to claim that people always know about all their beliefs, anymore then we want to claim that they believe everything that they could infer from their beliefs. Otherwise, we would end up with the following as a valid chain of inference:

John believes that snow is white.

John believes that John believes that snow is white.

John believes that John believes that John believes that snow is white.

John believes that John believes that John believes that John believes...

The second line is plausible enough, but the fourth line is weird, and if we continued the 500th line would be impossible to read, let alone believe. Introspection is like inference: it is something pecple do on purpose, and they do as much of it as they need for the problem at hand.

Many beliefs are the result of perception. People make inferences like the following:

John looked at a piece of paper with a number written on it.

John knew what number was written on the paper.

I stressed above that many beliefs arise from a deliberate effort of thinking. If we say that inference and introspection happen automatically, we get into trouble because these processes take beliefs as input and produce new beliefs as output. Therefore their output can be used as input for more introspection and inference, and if the process runs on automatically we might get an infinite set of beliefs. This problem does not occur with perception, because its input is not old beliefs, but physical events in the external world. Therefore no problem arises if we claim that perception creates new beliefs automatically. And this seems to be true. If someone sneaks up behind your back and blows a bugle in your ear, you'll notice it whether you want to or not.

6.3 The Situation Theory

Robert Moore's dissertation [13] uses a theory of belief based on Hintikka's possible worlds theory [6]. Moore had the ingenious idea of replacing Hintikka's possible worlds with the situations of McCarthy's situation calculus. Recall that the situation calculus is a technique for reasoning about actions. It introduces entities called situations, such that an object can have different properties in different situations, and at each instant of time the world is in exactly one situation. Since the properties of objects vary from situation to situation, a sentence can be true in one situation and false in another. Also, a description like "Bill's phone number" can denote different objects in different situations. One describes an action as a relation over situations. If this relation holds between situations s1 and s2, you can perform the action at any instant when the world is in situation s1, and if you do the world will be in situation s2 at the next instant. Moore dealt with knowledge only, but I will consider a natural extension of his theory to belief.

Moore proposed to represent an agent's beliefs as a set of situations, which I will call the agent's alternatives. If situation s is one of the agent's alternatives, then the agent's beliefs do not rule out the possibility that the current situation is s. In other words, for all he knows the world might be in situation s. Thus if the agent knows everything about the current situation, his set of alternatives contains only the actual situation. If he knows nothing at all his set of alternatives contains every situation. The more the agent learns, the more situations he rules out and the fewer his alternatives.

An agent believes that P if P is true in all of his alternatives. This explains at once why substitution of equals fails inside the scope of "believe". If John believes that Mary's number is 444-1212, then Mary's number is 444-1212 in all of his alternatives. If Bill's number is Mary's number, then Bill's number is Mary's number in the actual situation, and so Bill's number is 444-1212 in the actual situation. Still John's alternatives might include situations in which Bill and Mary have different numbers, and in these alternatives Bill's number is not 444-1212. So John does not necessarily believe that Bill's number is 444-1212.

This theory will also handle the first "knowing what" example. Moore says that an agent knows what X is if X is the same object in all of the agent's alternatives. That is, the agent's beliefs rule out all but one value of X. If the agent knows that Mary's number is 444-1212, then Mary's number is 444-1212 in all the agent's alternatives. Surely 444-1212 is the same number in all situations. That number is Mary's phone number in all of the agent's alternatives, so the agent knows what Mary's number is. On the other hand, suppose the agent knows only that Mary's number is Bill's number. Bill might have different phone numbers in different situations, so there need not be any one object that is Mary's number in all of the agent's alternatives. Again we get the right prediction.

Moore goes on to say that actions take arguments. For example, the action of dialing a phone number takes one argument, the number to be dialed. An agent knows how tc perform an action only if he knows what the action's arguments are. Then it follows that an agent knows how to dial Mary's number if he knows that Mary's number is 444-1212, but not if he only knows that Mary's number is Bill's number.

I claim that the situation theory of belief is wrong, and that a very different approach is needed (this is also Moore's current view - see [14]). The first criticism is that it makes false predictions about "knowing what". We don't say that you know what Mary's number is if you know that her number is equal to six times thirty-one squared. Yet six times thirty-one squared is surely the same number in every situation. In this case Mary's phone number is the same number in all of the agent's alternatives, yet he still doesn't know what her phone number is. Also, according to the situation theory whether an agent knows what X is depends only on the agent's alternatives. But we have seen that it can depend also on what the agent wants to do with the knowledge. If you want to put Mary in touch with John, and you know that John is standing next to you, you claim that you know where John is. If you want to direct John back to his hotel, and you know that he is standing next to you, you must learn more before you can claim to know where he is.

Suppose the agent believes that P, and P entails Q. Then P is true in all of the agent's alternatives and since P entails Q, Q is true in all of the agent's alternatives. That is, the agent believes Q. So in the situation theory an agent believes everything that follows logically from his beliefs. If the math professor in our previous example uses the situation theory to reason about his student's beliefs, he will conclude that they believe that every set can be well-ordered as soon as they know the axioms of set theory. There is a similar problem about introspection - as soon as an agent believes P he believes that he believes that he believes..., and so on forever.

This problem is not surprising in a theory that talks about beliefs, but not about the reasoning that creates beliefs. There may well be an infinite set of beliefs that an agent could infer, given arbitrary time and scratch paper. But at any time only a finite number have actually been inferred. If we say nothing about the inference that creates beliefs we can't distinguish between those that are easy to infer and those that take a long time. Then it's no wonder if we end up with a theory saying that everything is inferred in zero time. I conclude that the situation theory of belief is on the wrong track. We need a theory that describes the inferences that create beliefs.

6.4 The Syntactic Theory

6.4.1 A Robot and His Beliefs

I have described some of the data that a theory of belief and knowledge must handle, and how Moore fared with the situation theory of belief. Now I consider the syntactic theory. First comes a statement of the theory in

English, then the tools needed to formalize it, and then a series of example inferences.

I propose to take very seriously the idea that people are like computers. The agents in my theory look a lot like Von Neumann machines. Not that people are really like Von Neumann machines; rather common sense does not tell us about the massive parallelism and other non-Von Neumann things that go on in our heads. Let us imagine a simple robot, and build a theory that describes his beliefs. We will see that this theory can handle all of the given problems as well as the situation theory, and some of them better.

If we want to write a program that believes that snow is white, we devise a knowledge representation in which we can assert that snow is white - for example, by writing "(white snow)". Then we add this expression to a collection of expressions that are supposed to represent the program's beliefs. This practice suggests a theory: that beliefs are expressions of a knowledge representation language. This is the syntactic theory of belief. It appears now and again in the literature of philosophy - see [7], [3], and [10]. McCarthy [11] was the first AI worker to advocate this theory. Moore and Hendrix [14] argued that the syntactic theory can solve many philosophical problems about belief.

Men, machines and Martians can use very different internal languages to represent the same belief. I propose to ignore this possibility, and assume that all agents use the same representation for every belief. Our robot assumes that everybody else represents beliefs exactly as he does, and he ignores the difference between a belief and his representation of that belief.

Konolige [8] was the first to formalize this version of the syntactic theory. His treatment differs from mine in several important ways, which I will note as I come to them.

Suppose John believes that snow is white. The robot thinks that John's representation of this belief is the same as the robot's representation: the expression "(white snow)". The robot also thinks that the representation is the belief. It forms a name for the representation by putting quotation marks around it. So it represents the fact that John believes snow is white by an expression roughly like this.

(believe John "(white snow)")

The first argument of "believe" is the name of a man. The second argument is the name of an expression. To formalize the syntactic theory, one must assign names to expressions. That is, one must devise a system of quotation.

6.4.2 Formalizing the Syntactic Theory

I use predicate calculus with the following logical symbols:

(-> p q) - material implication
(& p q) - conjunction
(V p q) - disjunction
(~ p) - negation
(all x p) - universal quantification
(some x p) - existential quantification

This is the official notation; often I drop parentheses and use connectives as infix operators. A few predicates, like "<" and "=", will also be used as infix operators.

The beliefs of our hypothetical robot are sentences of a first-order logic extended with quotation. These beliefs need not be stored explicitly, but the robot must be able to find out whether he believes a given sentence or not in constant time by a standard retrieval algorithm. We do not say that you believe something if you can infer it after ten minutes of puzzling. All the beliefs are sentences of a single language L. When the robot forms beliefs about its own beliefs, those beliefs must be sentences of L that talk about sentences of L. This is a bit surprising. We are used to talking about an object language L1 by using a meta-language L2, where L1 and L2 are distinct. Why not stick to this method? Since the robot can form beliefs about beliefs about beliefs... up to any finite depth, we could set no limit to the number of meta-languages needed, but that is quite OK. If we follow this plan no language can ever talk about itself, but the robot can always form beliefs about his beliefs by going one step further in the hierarchy. Konolige used such a hierarchy of meta-languages in his formalization of the syntactic theory.

This plan will not work, because it forbids any belief to refer to itself. A belief can refer only to beliefs in languages lower in the hierarchy. In fact beliefs do refer to themselves. For example, a human might notice that he never forgets anything that interests him strongly. Suppose this belief interests him strongly; then it refers to itself, and quite likely makes a true assertion about itself. Or suppose the robot uses a pattern-matcher to retrieve beliefs from memory. It will need a belief describing the pattern-matcher, and this belief can be retrieved by pattern-matching like any other. Thus it says of itself "I can be retrieved by using such-and-such a pattern". There is nothing paradoxical or even unusual going on here. The point is important, because the decision to use a single self-describing language will involve us in the paradoxes of self-reference. One can avoid these paradoxes, but it is not easy.

One way to assign names to sentences is to let sentences be their own names. Then we could represent the fact that John believes snow is white by writing

(believe John (white snow))

This might be a good system, but it is impossible in first-order logic. Sentences denote truth values in first-order logic, they do not denote themselves. We must look farther for a quotation mechanism that will fit into first-order logic.

In English we form the name of a sentence by writing quotation marks around the sentence. Thus the expression

"Snow is white."

denotes the sentence

Snow is white.

If we adopt this scheme in our formal language we could represent the fact that John believes snow is white by writing

(believe John "(white snow)")

We can fit this scheme into first-order logic by saying that quoted expressions are constants that denote sentences. Yet this idea is not good enough, because it will not allow us to represent the fact that John knows what Mary's phone number is. We observed above that John knows what Mary's number is if he knows that Mary's number is n, where n is an Arabic numeral. We might try to represent this by writing

```
1
(some n (know John "(= (PhoneNumber Mary) n)")
        &
        (IsArabic n)
)
```

But this will not do. By definition of quotation marks, the second argument of the predicate letter "know" denotes the wff

(= (PhoneNumber Mary) n)

This is true no matter what the variable "n" is bound to. So the quantifier "some" does nothing, and (1) means the same as (2).

```
2
  (know John "(= (PhoneNumber Mary) n)")
& (some n (IsArabic n))
```

We need a quotation system that allows us to embed non-quoted expressions in quoted expressions. Then we can represent the fact we tried to represent with (1).

Instead of using a quotation mark that applies to whole expressions, let us quote the individual symbols. If we put the character ' in front of each symbol that we want quoted, we can write

```
3
(some n (know John ('= ('PhoneNumber 'Mary) n))
        &
        (IsArabic n)
)
```

to represent the fact that (1) fails to express. All the symbols in the second argument of "know" are quoted, except for the variable "n" which is bound by the quantifier in the ordinary way. If we can fit this quotation scheme into first-order logic, we can formalize the syntactic theory.

The problem is to assign denotations to the quoted symbols so that sentences like (3) will have the intended meanings, given the usual semantic rules of first-order logic. To each constant of our language we assign a name, formed by appending the character ' to that constant. Thus if "Mary" is a constant and denotes a woman, "'Mary" is a constant and denotes the constant "Mary". To each variable we assign a name in the same way. If "x" is a variable, then "'x" is a constant that denotes the variable "x".

Now consider the symbols that take arguments - function letters, predicate letters, connectives and quantifiers. These symbols are called functors. The term "(& P Q)" consists of the functor "&" and its arguments "P" and "Q". If "F" is a functor of n arguments, then "'F" is a function letter. It denotes the function that maps n expressions e1 ... en to the expression with functor "F" and arguments e1 ... en. For example, the function letter "'&" denotes the function that maps wffs w1 and w2 to the wff with functor "&" and arguments w1 and w2 - which is the conjunction of w1 and w2. The function letter "'~" denotes the function that maps a wff to its negation, and so on.

If the variable "n" denotes the arabic numeral "5766", then the term

('= ('PhoneNumber 'Mary) n)

should denote the sentence

(= (PhoneNumber Mary) 5766)

The function letter "'PhoneNumber" denotes the function that maps a term t to the term with function letter "PhoneNumber" and argument t. The constant "'Mary" denotes the constant "Mary". So the term

('PhoneNumber 'Mary)

denotes the term with function letter "PhoneNumber" and argument "Mary", which is

(PhoneNumber Mary)

The function letter "'=" denotes the function that maps terms t1 and t2 to the wff with predicate letter "=" and arguments t1 and t2. So the term

('= ('PhoneNumber 'Mary) n)

denotes the wff with function letter "=" and arguments "(PhoneNumber Mary)" and "5766", which is

(= (PhoneNumber Mary) 5766)

And that is the answer we want.

So if the robot knows what Mary's phone number is, it can represent this fact by the sentence

```
(some n (know Me ('= ('PhoneNumber 'Mary) n))
        &
        (IsArabic n)
)
```

The constant "Me" is the robot's selfname - the robot's usual name for itself. "know" is an ordinary predicate letter - not a special modal operator as in Hintikka. The model theory of our language contains no special rules for interpreting the predicate "know".

On the other hand, suppose that the robot only knows that Mary has a phone number. We represent this as

(know Me ('some 'n ('= ('PhoneNumber 'Mary) 'n)))

In this case the existential quantifier is inside the quotation mark.

The term

4 ('= ('PhoneNumber 'Mary) '5766)

includes the quote name of the arabic numeral for Mary's phone number. The term

5 ('= ('PhoneNumber 'Mary) n)

has a variable in the same position. (4) is the quote name of a wff, but (5) is a wff schema. The quote name of a wff includes a quote name for every term in that wff. A wff schema is like the quote name of a wff, except that variables can appear in place of the quote names of terms. A wff w is called an instance of a wff schema s if for some assignment of values to the free variables in s, s denotes w. For example, if the variable "n" is assigned the value "5766", then (5) denotes

6 (= (PhoneNumber Mary) 5766)

So the sentence (6) is an instance of the wff schema (5).

Writing a quotation mark in front of every functor is a nuisance, so we abbreviate by putting the quotation mark in front of a whole expression. Thus

"'(PhoneNumber Mary)" abbreviates "('PhoneNumber 'Mary)". I use infix notation for the connective "&", but never for the quoted function letter "'&". People don't usually use infix notation for function letters, and I want to emphasize that quoted function letters really are function letters. They obey every syntactic and semantic rule that governs function letters in first-order logic. In particular, we apply quotation marks to quoted function letters like any other function letter. Thus "''Superman" denotes the quoted constant "'Superman", which denotes the quoteless constant "Superman", which denotes the man from Krypton.

We also need the function letter "quote", which denotes the function that maps an expression to its quote name. This function maps the wff "(white snow)" to the term "('white 'snow)", for example. So we write

(quote ('white 'snow)) = (''white ''snow)

The argument of "quote" is a term that denotes the wff "(white snow)". The right-hand argument of the equals sign denotes the term "('white 'snow)". This sentence says that the quote name of "(white snow)" is "('white 'snow)" - which is true.

The difference between the quotation mark ' and the function letter "quote" is this. If "v" is a variable, then "'v" is a constant that denotes that variable. "(quote v)" is a term in which the variable "v" is free, and its value depends on the value of "v". If the value of "v" is the constant "Superman", then "(quote v)" will denote the quote name of the constant "Superman", which is "'Superman".

6.5 Applying the Syntactic Theory

I have now explained the syntactic theory and the machinery used to formalize it. The next task is to apply the formalized theory to the examples described in Section 6.2

6.5.1 Observation

The robot forms new beliefs by observing the external world and his own internal state. The world is always changing, so the robot needs a theory of time, and it must be able to perceive the passage of time.

6.5.1.1 Time

Time is a set of instants totally ordered by <. If instant i precedes instant j there is an interval whose lower endpoint is i and whose upper endpoint is j. It contains the instants that are later than i and earlier than j. The lower endpoint of interval I is −I, and its upper endpoint is +I. Nearly all properties of objects hold during intervals. In particular, we write (believe A S I) to indicate that agent A believes sentence S during interval I. Actions happen during intervals. Thus we write (puton Robot A B I) to indicate that the robot puts block A on block B during interval I.

We can define the order relations between intervals in terms of the < relation between their endpoints. For example, interval I is before interval J if the upper endpoint of I is before the lower endpoint of J: +I < −J. Interval I meets interval J if the upper endpoint of I is the lower endpoint of J: +I = −J.

The robot has sensors - devices that detect events in the outside world and produce descriptions of those events in the robot's internal language. The sensors accept physical events as input and produce sentences as output. These sentences become beliefs. A belief created by perception must note the time of the perception. For suppose the robot receives the same message from his sensors at two different times - hears two rifle shots in succession, for example. If the beliefs created by these two perceptions do not mention the times at which the perceptions happened, they will be identical. Then the robot's collection of beliefs will be the same as if it had heard only one shot.

Therefore the robot will need names for intervals of time. These names are constants of the internal language called time stamps. If the robot hears the doorbell ring during interval I, it creates a time stamp for interval I - say "Interval101". Then it adds to its beliefs the sentence

(ringing Interval101)

which says that there is a ringing sound during Interval101. The robot automatically records every perception, and also other events such as inferences and commands to the effectors. Whenever it records such an event it creates a time stamp for the interval when the event happened. It uses that time stamp to name the interval in the belief that records the event.

A time stamp is a useful name for an interval because the robot keeps records of the lengths and order of intervals, and uses time stamps to name the intervals in those records. If the robot creates a time stamp "Interval53" for an interval J, then as soon as interval J is over the robot forms a belief that records its length. This estimate of the interval's length need not be accurate. People can't tell a minute from fifty seconds without a watch, but they can tell a minute from a second. The robot can get by with rough estimates too. Let us choose a small unit of time and approximate the lengths of intervals with whole numbers of units. Then if J is 30 units long, there is an interval K such that J meets K and

(believe Robot '(= (length Interval53) 30) K)

This belief gives the length of the interval in units, using an arabic numeral to name the number of units. For any integer n, let (arabic n) be the arabic numeral that denotes n. So (arabic 2+2) = (arabic 4) = '4. Suppose the robot creates a time stamp t for an interval i whose length is n units. Then there is an interval j such that i meets j and

(believe Robot ('= ('length t) (arabic n)) j)

Setting t = 'Interval53, n = 30, j = K gives

(believe Robot ('= ('length 'Interval53) (arabic 30)) K)

Since (arabic 30) = '30, we have

(believe Robot ('= ('length 'Interval53) '30) K)

which is a notational variant of the last example.

The robot also records the order relations between intervals that have time stamps. To record the order relation between two intervals it is enough to record the order relations between their endpoints. Given intervals I,J we must record the order relations between -I and -J, -I and +J, +I and -J, +I and +J. Consider the first case. If i and j are intervals with time stamps t1,t2, the robot will record the order relation between +i and +j immediately after the later of the two instants. There are three cases to consider. If +i < +j there is an interval k whose lower endpoint is +j, and (believe Robot ('< ('+ t1) ('+ t2)) k)

If +i = +j there is an interval k whose lower endpoint is +i, and

(believe Robot ('= ('+ t1) ('+ t2)) k)

Finally, if +j < +i there is an interval k whose lower endpoint is +i, and

(believe Robot ('< ('+ t2) ('+ t1)) k)

So the robot always knows the order relations among all intervals that have been assigned time stamps. Thus the robot has a sense of time: if it remembers two perceptions it remembers which came first and how long they lasted. This particular axiomatization of the sense of time is crude, but it will do for our purposes. One could do a better job with the same formalism if necessary.

6.5.1.2 Perception

Certain physical events cause the robot's sensors to produce sentences that describe those events. Let us write (perceive Robot s i) to indicate that during interval i the robot's sensors produce the sentence s as a description of some event or state in the outside world. As an example, let us describe the robot's ability to read. The symbols we read and write are expressions of English, not expressions of the robot's internal language. Let us gloss over this distinction and pretend that expressions of the thought language can be written on paper, and the robot can read them.

Suppose that the robot's field of view is a rectangle, and the sensors use integer Cartesian coordinates to descxibe positions in the field of view. Let (written e x y i) indicate that the expression e is written down at coordinates (x,y) in the robot's field of view during interval i. If this is the case the robot's sensors will report it, using a quote name for the expression e, arabic numerals for the integers x and y, and a time stamp for the interval i. Suppose that e is an expression, x and y are coordinates, and i is an interval. If (written e x y i), there is a time stamp t for the interval i, and

(perceive Robot ('written (quote e) (arabic x) (arabic y) t) i)

Suppose that "(white snow)" is written at coordinates (150,150) in the robot's field of view during interval I. Then there is a time stamp for interval I, say "Interval99", and we have

```
(perceive Robot
          ('written (quote '(white snow))
                    (arabic 150)
                    (arabic 150)
                    'Interval99
          )
          I
)
```

Using (quote '(white snow)) = ''(white snow) and (arabic 150) = '150 gives

```
(perceive Robot
          ('written ''(white snow) '150 '150 'Interval99)
          I
)
```

The robot believes what its sensors tell it. That is, if it perceives a sentence s during interval i, there is an interval j such that i meets j and the robot believes s during j. In this case there is an interval K such that I meets K and

(believe Robot ('written ''(white snow) '150 '150 'Interval99) K)

6.5.1.3 Retrieving Beliefs From Memory

The robot acts by executing programs, and its programming language is quite conventional. There is a fixed set of registers. Just like a Von Neumann machine, the robot must bring a data structure into a register before it can operate on that data structure. Remembering a belief means bringing it from memory into a register. If the robot has Bill's phone number stored in its memory, but for some reason can't retrieve it, it cannot call Bill. It has no way to pass the phone number to its telephone dialing routine. This matches our intuitions about people: if you know Bill's phone number, but you can't remember it at the moment, then you can't call Bill.

The statements of the programming language are terms of the internal language, although they have no useful denotations. Considering them to be terms of the internal language is handy because we can then use quotation to name programs. The expressions of the programming language are terms of the internal language, and their values in the programming language are their denctations. Of course they are limited to terms whose values the agent can compute.

All the expressions of the internal language are data structures of the programming language. There are other data structures in the programming lenguage - lists of expressions, for example. Every data structure has a name in the internal language called its print name. The print names of expressions are just their quote names. The print name of the list (cons e nil) is ('cons (PrintName e) 'nil).

The robot uses a statement called the retrieve statement to retrieve beliefs from his memory. A retrieve statement has the form (retrieve r p c), where r is a register, p is a wff schema, and c is a wff. p is called the pattern and c is called the condition. Suppose the robot wants to retrieve a sentence that tells what John's phone number is. Such a sentence has the form

7 ('PhoneNumber 'John n)

The term n must be an arabic numeral:

8 (IsArabic n)

The robot can retrieve a sentence that tells what John's phone number is by executing a retrieve statement with pattern (7) and condition (8):

```
9 (retrieve R1
            ('PhoneNumber 'John n)
            (IsArabic n)
  )
```

A sentence s matches the pattern "('PhoneNumber 'John n)" and the condition "(IsArabic n)" if for some binding of the variable "n", "('PhoneNumber 'John n)" denotes s, and "(IsArabic n)" is true. For example, if "n" is bound to "5766", then "('PhoneNumber 'John n)" denotes "(PhoneNumber John 5766)" and "(IsArabic n)" is true. Therefore "(PhoneNumber John 5766)" matches the pattern "('PhoneNumber 'John n)" and the condition "(IsArabic n)". If a sentence matches the pattern "('PhoneNumber 'John n)" and the condition "(IsArabic n)", then it has the form ('PhoneNumber 'John n) for some arabic numeral n. That is, it tells what John's phone number is. So if the robot knows what John's phone number is, he can retrieve that knowledge by executing the statement (9).

In general, a sentence s matches pattern p and condition c if p is a wff schema and for some bindings of the free variables of p, p denotes s and c is true. Suppose the robot executes the statement ('retrieve r p c) in interval I, and the robot believes a sentence that matches pattern p and condition c. Then the retrieve statement returns a belief that matches the pattern and the condition. There may be several beliefs that match. If so any one of them might be returned. Register r is set to the belief that is returned. That is, there is an interval J such that I meets J and register r holds the returned belief during J. The retrieve statement allows the robot to search his memory.

6.5.1.4 Introspection

Now that we have a statement that searches the memory we can describe introspection very neatly. All we have to do is say that whenever an agent executes a statement he knows whether it returned a value, and if so what value. The agent can then find out whether he has a certain belief by trying to retrieve it. If he succeeds he will know this, and he can infer that he had the belief; if he fails he will know this also, and he can infer that he had no belief that matched the pattern and the condition.

Suppose, then, that the robot executes a statement s of the programming language during interval I, and it returns a value v. The value v is a data structure. The robot has a time stamp t for the interval I. There is an interval J such that I meets J, and during interval J the robot believes the sentence

('return (SelfName Robot) (quote s) (PrintName v) t)

This sentence says that the robot executed statement s during interval I, and it returned value v. The robot is named by his selfname, the interval by a time stamp, the statement by its quote name, and the returned value by its print name. If the robot executes the statement

(retrieve R1 ('PhoneNumber 'John n) (IsArabic n))

and it returns the sentence

(PhoneNumber John 5766)

he will believe

```
(returns Me
         '(retrieve R1 ('PhoneNumber 'John n) (IsArabic n))
         '(PhoneNumber John 5766)
         Interval432
)
```

assuming "Interval432" is the time stamp for interval I. The robot knows that if he executes a retrieve statement during any interval i, and it returns a sentence s, he believed s during i. So he can infer

(believe Me '(PhoneNumber John 5766) Interval432)

So if the robot believes that John's number is 5766, he can find out that he believes that John's number is 5766.

Suppose the robot executes a statement s of the programming language during interval I,and it returns no value. The robot has a time stamp t for the interval I. There is an interval J such that I meets J and during interval J the robot believes the sentence

('~ ('some 'x ('returns (SelfName Me) (quote s) 'x t)))

This sentence says that the robot executed statement s during interval I, and it returned no value.

Suppose the robot does not know what John's phone number is. That is, he has no belief of the form ('PhoneNumber 'John n), where n is an arabic numeral. If the robot executes the statement

```
(retrieve R1 ('PhoneNumber 'John n) (IsArabic n))
```

it will return no value. For only a belief of the form ('PhoneNumber 'John n), where n is an arabic numeral, would match the pattern and the condition. If this statement returns no value, the robot will believe the sentence

```
(~ (some x
    (returns Me
             '(retrieve R1 ('PhoneNumber 'John n) (IsArabic n))
             x
             Interval82
    )
))
```

This sentence says that the retrieve statement returned no value. The robot can now argue by contradiction: If I had a belief of the form ('PhoneNumber 'John n), where n was an arabic numeral, it would have matched the pattern "('PhoneNumber 'John n)" and the condition "(IsArabic n)". Then the retrieve statement would have returned a value. But the retrieve statement returned no value. Therefore I have no belief of the form ('PhoneNumber 'John n), where n is an arabic numeral. That is, I do not know John's phone number.

Most theories of belief include an axiom saying that if an agent believes that P, he believes that he believes that P. This theory has instead a general axiom of introspection. It says that if an agent executes a statement of his internal programming language, he knows what value it returned. This axiom allows us to show that if an agent believes that P he can easily discover that he believes that P. We use the same axiom to show that if the agent does not believe that P, he can discover that he does not believe that P. Also we can show that if the agent does not know what X is, he can discover that he does not know what X is - at least in some cases. Later we will find another use for this axiom of introspection.

6.5.2 Inference

Inference is another process that creates new beliefs. A.I. workers have often distinguished between data-driven and goal-driven inferences. The data-driven inferences happen whenever certain kinds of data are added to the data base. The goal-driven inferences happen when the robot is trying to prove certain kinds of theorems. Data-driven inferences must be limited in some way, because the robot can have only a finite number of beliefs. Breaking up conjunctions is a reasonable data-driven inference: if p & q is added to the belief base, p and q are added too. We can easily describe this with an axiom:

```
(believe Robot ('& p q) i)
-> (believe Robot p i) & (believe Robot q i)
```

The new beliefs formed by breaking up conjunctions could be added explicitly. They could also be added implicitly, by using a belief retrieval program that looks inside conjunctions. Such implementation questions are outside the scope of this theory.

6.5.2.1 What Do John's Beliefs Entail?

I turn now to the problem of predicting goal-driven inferences. I begin with the usual distinction between search space and search algorithm. To show that an agent will infer a certain belief if he tries to infer it, we must show that there is a path in his search space that leads to that belief, and that his search algorithm is powerful enough to find it. I consider first the problem of showing that the path exists - that is, the agent's beliefs entail the given sentence.

Suppose our knowledge of another agent's beliefs consists of a set of sentences of the form (believe agent (quote s) i) - that is, we have quote names for the sentences the other agent believes. Then by removing the quotation marks we can reconstruct the exact sentences that the other agent believes. We build a data base, separate from our collection of beliefs, containing the sentences that the other agent believes. Any theorem that we can prove using only the sentences in this data base follows from the other agent's beliefs. This is an old idea. Creary [2] was the first to point out that we can combine this kind of reasoning with the use of quotation to represent beliefs.

This method is not sufficient to handle the following inference.

John knows what Mary's phone number is.
John knows that Mary's phone number is the same as Bill's.

John knows what Bill's phone number is.

In our notation the first sentence becomes

```
(know John
      ('= '(PhoneNumber Mary) (arabic (PhoneNumber Mary)))
      I
)
```

We cannot reconstruct the sentence that John believes from this description, because the description doesn't tell us which arabic numeral appears in John's belief. So we can't build a data base containing John's beliefs.

Konolige [8] suggested a solution to this problem. Instead of using simulation, he proposed to describe the proof rules of the other agent's language, and use this description to show that a theorem can be proved from another agent's beliefs. For example, we might describe the rule of Modus Ponens by writing

(all p q (ModusPonens p ('-> p q) q))

This axiom says that p and ('-> p q) entail q by the rule of Modus Ponens. We do not need quote names for the wffs p and q to use this axiom - any names at all will do. If we follow Konolige the lack of quote names for John's beliefs creates no special problem.

The difficulty with Konolige's proposal is that it leads to very long proofs. Suppose we try to find the conclusion of an n-step proof using axioms that describe the proof rules. For each step of the original proof we must build a short proof, which shows that that step produces a certain conclusion. Suppose the average length of these proofs is p steps. Then the proof that our n-step proof has a certain conclusion will involve p X n steps.

Now consider what happens when we nest this kind of reasoning. Suppose John knows that Mary knows his phone number. Then John expects Mary to know how to call him. Suppose we apply Konolige's technique to this problem. There is an n-step proof whose premisses are among Mary's beliefs, and whose conclusion says that Mary can call Bill by dialing a certain number. To show that the proof has this conclusion, John must build a proof of p X n steps. To show that John can build this proof, we must build a proof of p x (p X n) steps. The size of the proofs grows exponentially with the depth of nesting. This is clearly intolerable.

There is an obvious way out of this problem, although it is not trivial to show that it is correct. We pick a new constant, say "C", and use it to stand for the arabic numeral that John uses to name Mary's phone number. Then we can build a data base that approximates John's beliefs. It will contain the sentences

```
(= (PhoneNumber Mary) C)
(= (PhoneNumber Mary) (PhoneNumber Bill))
```

from which we can infer

```
(= (PhoneNumber Bill) C)
```

Since "C" stands for an arabic numeral, John can infer a sentence of the form ('= ('PhoneNumber 'Bill) n), where n is an arabic numeral. That is, John can figure out what Bill's phone number is.

Let us state the argument more precisely. If we were to go through the proof we have just built, and replace the constant "C" with the arabic numeral that appears in John's belief about Mary's phone number, the result would be a new proof. John believes the premisses of this proof, and its conclusion gives an rabic numeral for Bill's phone number. So there is a way to prove from John's beliefs a theorem that gives an arabic numeral for Bill's phone number. The crucial assumption here is that if we go through the proof and replace "C" with another constant, the result is still a proof. Given a suitable version of first-order logic, one can show that if we take any proof and replace all occurrences of a constant with a closed term, the result is still a proof. This theorem justifies the use of a new constant to represent an unknown term that appears in another agent's beliefs. It allows us to prove the correctness of an axiom schema called the Reflection Schema, which does the kind of reasoning that we have informally described.

6.5.2.2 The Reflection Schema

Consider first the simple case of the Reflection Schema, in which we have the quote names of the other agent's beliefs. The proof to be reflected consists of a single step. The rule of Substitution Of Equals is applied to the premisses

```
(= (PhoneNumber Mary) 5766)
(= (PhoneNumber Bill) (PhoneNumber Mary))
```

producing the conclusion

```
(= (PhoneNumber Bill) 5766)
```

A proof is formed by starting with wffs called premisses and repeatedly applying proof rules. Every proof has a print name, which includes the quote names of all the premisses of the proof. Given the print name of a proof one can easily reconstruct that proof, just as one can reconstruct a sentence from its quote name. The print name of the proof just given is

```
(EqSubst '(= (PhoneNumber Mary) 5766)
          '(= (PhoneNumber Bill) (PhoneNumber Mary))
          '(= (PhoneNumber Bill) 5766)
)
```

(IsProof p) means that p is a correct proof. If p is a correct proof, the sentence

('IsProof (PrintName p))

is an instance of the Reflection Schema. For the proof given above we have the instance

```
10 (IsProof (EqSubst '(= (PhoneNumber Mary) 5766)
                     '(= (PhoneNumber Bill) (PhoneNumber Mary))
                     '(= (PhoneNumber Bill) 5766)
            )
   )
```

If John believes the premisses of this proof he can infer its conclusion – that is, he can infer that Bill's number is 5766.

It is easy to implement this schema. The implementation is a program that takes a sentence as input and decides whether it is an instance of the Reflection Schema. The input sentence contains the print names of a proof and its conclusion. From the print names the program reconstructs the proof and the conclusion. Then it calls the programs that implement the other proof rules to decide whether the proof is correct. If it is, the input sentence is an instance of the Reflection Schema.

Suppose John knows Mary's phone number. Then he believes the sentence

('= '(PhoneNumber Mary) (arabic (PhoneNumber Mary)))

If he also believes that Bill's phone number is the same as Mary's, he believes the sentence

'(= (PhoneNumber Bill) (PhoneNumber Mary))

By Substitution Of Equals he can infer

('= '(PhoneNumber Bill) (arabic (PhoneNumber Mary))

The version of the Reflection Schema that we have just seen is not strong enough to prove this. It demands the quote names of the sentences in the proof to be reflected, and we do not have the quote name of the arabic numeral for Mary's phone number. We need a stronger Reflection Schema, which includes the following instance.

```
11
(all x
  (ClosedTerm x)
  -> (IsProof
       (EqSubst ('= '(PhoneNumber Mary) x)
                ('= '(PhoneNumber Bill) '(PhoneNumber Mary))
                ('= '(PhoneNumber Bill) x)
       )
     )
)
```

Since an arabic numeral is a closed term, we infer

```
(IsProof
  (EqSubst ('= '(PhoneNumber Mary) (arabic (PhoneNumber Mary)))
           ('= '(PhoneNumber Bill) '(PhoneNumber Mary))
           ('= '(PhoneNumber Bill) (arabic (PhoneNumber Mary)))
  )
)
```

And this is the desired conclusion.

The argument of the predicate letter "IsProof" in (11) is called a proof schema. A proof schema is like the print name of a proof, except that a variable can appear instead of the quote name of a term. That is, a proof schema is to the print name of a proof what a wff schema is to the quote name of a wff. The argument of "IsProof" in (10) is the print name of a proof. It gives the quote name "'5766" of the arabic numeral for Mary's phone number. The argument of "IsProof" in (11) is a proof schema. The variable "x" appears in place of the quote name "'5766".

Implementing this version of the Reflection Schema is not as easy as implementing the first version. The first version reconstructed the proof to be reflected from its print name, and then called the other proof rules to decide whether the proof was correct. The new version gets only a proof schema, which stands for a whole class of proofs called instances of the schema. A proof p is an instance of a proof schema s if for some bindings of the free variables of s, s denotes p. The proof named in (10) is an instance of the proof schema that appears in (11). It is formed by binding the variable "x" to the term "5766". An instance of the Reflection Schema is true only if all instances of its proof schema are correct proofs.

The solution is to form a proof called the typical instance of the proof schema. Every instance of the scheme can be formed by substituting closed terms for constants in the typical instance. If the typical instance is a correct proof, then since substitution maps proofs to proofs, all the instances are correct proofs.

The typical instance of a proof schema is its denotation in an environment that binds its free variables to new constants. For example, if we bind the variable "x" to the new constant "C", then the proof schema in (11) denotes a proof whose premisses are

```
(= (PhoneNumber Mary) C)
(= (PhoneNumber Mary) (PhoneNumber Bill))
```

Its conclusion is

```
(= (PhoneNumber Bill C)
```

and the rule used is Substitution Of Equals. Since this proof is correct, all instances of the proof schema in (11) are correct. Therefore (11) is a true sentence (in the intended model). This is a rough explanation of the Reflection Schema, omitting many complications.

We have shown how to make inferences by simulation from atomic sentences like

```
(believe John
         ('= '(PhoneNumber Mary) (arabic (PhoneNumber Mary)))
         1
)
```

and from conjunctions of such sentences. What about the other connectives and quantifiers? We ought to be able to make inferences from disjunctions and negations of sentences about belief, and from universally or existentially quantified statements about belief. Here we see the value of adding the Reflection Schema to a first-order logic. We can handle negation, disjunction and quantification without adding any more rules or axioms.

Consider negation. If John believes that Mary's number is 5766, and he does not believe that Bill's number is 5766, he must not believe that Bill's number is Mary's number (assuming that he can make trivial inferences). First-order logic allows us to argue by contradiction: to prove ~p by assuming p and proving a contradiction. So assume

```
(believe John '(= (PhoneNumler Mary) (PhoneNumber Bill)) 1)
```

and

```
(believe John ('= ('PhoneNumber 'Mary) '5766) 1)
```

We can prove by simulation that there is a one-step argument from these beliefs of John's to the conclusion

```
(= (PhoneNumber Bill) 5766)
```

So if John can find this one-step argument, he believes that Bill's number is 5766. This contradicts the assumption that he has no such belief, so we can conclude that John does not believe that Bill's number is Mary's number.

Suppose John believes that all millionaires are happy, and he either believes that Bill is a millionaire or that Bob is a millionaire - we don't know which.

```
  (believe John ('all 'x ('-> ('millionaire 'x) ('happy 'x)) I)

  (believe John ('millionaire 'Bill) I)
V (believe John ('millionaire 'Bob) I)
```

We should be able to prove that he believes someone is happy (again assuming he can make trivial inferences).

```
(believe John ('some 'x ('happy 'x)) I)
```

First-order logic allows us to argue by cases - to prove p from q V r by proving p from q and proving p from r. Assume first that John believes that Bill is a millionaire. Then we can build a data base that represents his beliefs, and it looks like this:

```
  (all x (millionaire x) -> (happy x))
  (millionaire Bill)
```

In this data base we can easily infer that someone is happy:

```
(some x (happy x))
```

So John believes that someone is happy. Now suppose John believes that Bob is a millionaire. We can prove again that John believes someone is happy. Since John either believes that Bill is a millionaire or that Bob is a millionaire, it follows that he believes someone is happy.

Suppose John knows every millionaire by name. That is, for each millionaire he be·ie·es that N is a millionaire, where N is the millionaire's personal name. Let (name x) be x's personal name, for any person x. (name John Smith) is "John Smith", and so on. Then we can describe John's exhaustive knowledge of millionaires by writing:

```
(all x (millionaire x)
       -> (believe John ('millionaire (name x)) 1))
```

John also believes that every millionaire is happy.

```
(believe John '(all x (-> (millionaire x) (happy x))) 1)
```

We should be able to infer that for each millionaire, John believes that he is happy.

```
(all x (millionaire x) -> (believe John ('happy (name x)) 1))
```

First-order logic allows us to prove that everything has a certain property by proving that an arbitrary object has that property. In the first-order proof system used here, free variables represent arbitrary objects. So let z be an arbitrary object, and suppose z is a millionaire. Since John knows every millionairc by name, we have

```
(believe John ('millionaire (name z)) 1)
```

We choose the constant "C" to stand for the unknown term (name z). Then the data base that represents John's beliefs contains the sentences

```
(millionaire C)
(all x (millionaire x) -> (happy x))
```

These sentences entail

```
(happy C)
```

We conclude

(believe John ('happy (name z)) 1)

But since z represents an arbitrary millionaire, we have

(all x (millionaire x) -> (believe John ('happy (name x)) 1))

which was to be proved. The treatment of existentially quantified sentences about belief is similar.

By adding the rule of Reflection to a first-order logic, and proving that it is correct, we gain two advantages. Because we have proved the rule correct, we can rule out the possibility of bugs caused by bad interactions between Reflection and some obscure feature of the logic. Because the rule is part of a system that represents negation, disjunction and quantification correctly and completely, no extra work is needed to handle negations, disjunctions and quantifications of statements about belief. Imagine what would have happened if we had first written a rough description of our inference rules in English, then written 30 pages of LISP to implement them, and then started "maintaining" the code so that it changed once a week. How would one show that replacing constants with closed terms maps proofs to proofs? The kind of work we have just done is possible only if the rules of inference are set down plainly. These considerations prove nothing about the merits of frames vs. semantic nets vs. logic. They do indicate that an if an AI system is going to use reflection to reason about belief, its inference rules must be made explicit, not hidden in the code.

6.5.2.3 What Can John Infer from His Beliefs?

The rule of Reflection allows us to show that an agent's beliefs entail a sentence. To show that an agent will actually infer that sentence, we need to show also that his theorem prover is powerful enough to find a proof. The agent calls his theorem prover by executing a prove statement. This statement has the form ('prove r p c), where r is a register, p is a pattern, and c is a condition. When the agent executes this statement his theorem prover tries to prove a sentence that matches the pattern and the condition. If it succeeds, the prove statement returns that sentence and puts it in register r. Also the sentence is added to the agent's beliefs.

This leaves the crucial questions: can the theorem prover prove a sentence that matches the pattern and the condition? If it can, which one will it prove? Creary [2] offered a simple answer. If agent A can prove by simulation that agent B's beliefs entail P, then agent B can prove P. This is very different from saying that if agent B's beliefs entail P, agent B can prove P. It is quite possible that B's beliefs entail P but A cannot prove this fact.

Agent A predicts the behavior of agent B's theorem prover by making an empirical observation of the behavior of his own theorem prover. This involves the assumption that agent A is not much brighter than agent B - an assumption that is reasonable in most common sense contexts, though not when A is a math teacher and B is a student. Agent A can answer the question "Which theorem will agent B prove?" by similar reasoning. Perhaps there are many theorems entailed by agent B's beliefs that would match the pattern and condition that B gave to his theorem prover. If A has shown that a particular theorem entailed by B's beliefs will match the pattern and condition, he can assume that this theorem is an obvious answer, and it is the one B will prove.

This is not so convincing as the last assumption, but it is the same principle - A is predicting the behavior of B's theorem prover by observing the behavior of his own theorem prover. Agent A can even make a rough estimate of the time it will take for B to prove the theorem. it should be no more than the time it took A to simulate B's reasoning. We already have an axiom of introspection, which says that when agent A executes a statement of his programming language he knows what it value it returned. So if agent A executes the prove statement he knows what theorem he proved. Also he has a time stamp for the interval in which he executed the prove statement, so he knows how long it took.

Suppose then that A and B are two agents, and the following conditions hold.

i. Agent B executes the statement ('prove r p c) during an interval i, and sentence s matches the pattern p and condition c.
ii. Agent B's beliefs entail sentence s.
iii. Agent A has proved during an interval j that agent B's beliefs entail s.

Then agent B's execution of ('prove r p c) returns the value s, and interval i is no longer than interval j. The first condition says that agent B is trying to prove sentence s, or one like it. The second condition says that there is a proof of s from agent B's beliefs. The third condition says that agent A has found such a proof, so it is not too difficult.

Now we can do the following example from part 1.

John knows that Mary's number is 5766.
John knows that Mary's number is the same as Bill's number.
John is trying to figure out what what Bill's number is.

John will infer that Bill's number is 5766.

We represent the statement that John is trying to figure out what P is by saying that John has called his thecrem prover and asked it to prove a sentence that tells what P is. In this case, John wants his theorem prover to prove a sentence that tells what Bill's phone number is. Using a new constant C to stand for the arabic numeral for Bill's phone number, the robot can simulate John's reasoning. Having simulated the reasoning, it infers that John can do the same reasoning, so he will figure out what Bill's number is.

An agent figures out what inferences another agent can make by simulating his reasoning. If the other agent's beliefs include terms that are unknown to the simulator, he must use an approximation of the other agent's beliefs in his simulation. The simulator introduces new constants to represent the unknown terms in the other agent's beliefs. By noting the time that it took him to simulate the other agent's reasoning, the simulator judges how hard it will be for the other agent to find the same line of reasoning. The simulator does not have or need a theory that explains why one line of reasoning is harder to find than another. He uses empirical observations of the behavior of his own theorem prover to predict the behavior of another agent's theorem prover. So we have a theory of belief that talks about the processes that create beliefs.

6.5.3 Knowing What

John kn.ws what Mary's phone number is if he knows that Mary's number is n, where n is an arabic numeral. We represent this as

```
(some n    (know John ('= '(PhoneNumber Mary) n))
        & (IsArabic n)
)
```

We can give a similar treatment of other "knowing what" examples. An English teacher would say that a student knows who the author of "Hamlet" is if he knows that the author of "Hamlet" is n, where n is a personal name. We can represent this as

```
(some n    (know student ('= '(author Hamlet) n)
        & (IsPersonalName n)
)
```

Since "Shakespeare" is a personal name, the student knows who the author of "Hamlet" is if he knows that the author of "Hamlet" is Shakespeare. When we say that someone knows what X is, we mean that he knows that X is n, where n is a name or description having some property P. In the first case, P is the property of being an arabic numeral. In the second case, P is the property of being a personal name. Kaplan [7] suggested this approacn.

As we saw in the example of being lost in the city, the property P depends on context. In that example, the agent wanted to use the name n to accomplish a task. First the task was to get back to the hotel, and he wanted n to be something like "five blocks north of the hotel on High Street". Then he switched to a new task: helping Mary to find John. For this task the term "here" described John's location quite well. This example suggests that John knows what X is if he knows that X is n, where the name or description n contains the information needed for the task at hand.

That would certainly explain why knowing an arabic numeral for Mary's phone number counts as knowing what her number is. The arabic numeral allows us to call Mary, which is what phone numbers are for. Or suppose John tells me that he lives in the grey house across the street from the Star Market on Park Avenue. Then if I know how to get to the Star Market I can get to John's house. I could also claim that I know where John lives, even if I have never seen his house. This example fits the proposal nicely.

According to Konolige I know where John lives only if I have a standard name for John's house - one that denotes the same house in all possible worlds. Certainly "the grey house across from the Star Market" does not denote the same house in all possible worlds, so Konolige predicts that in this case I do not know where John lives. Since Konolige does not suggest that the set of possible worlds under consideration can vary with context, he does not allow context to determine whether knowing that X is n entails knowing what X is. Moore's proposal is that I know what X is if X is the same object in all my alternatives. This is different from Konolige's idea, because my alternatives are a small subset of the set of all situations. It still does not explain how I can know where John lives when I have never seen his house, and can only describe it as "the grey one across from the Star Market".

Alas, there are plenty of examples where my proposal fails. Often there is no particular task at hand. In a discussion of politics I may ask "Do you know who the Saudi oil minister is?". Presumably I want his personal name, but there is no obvious task to be accomplished with this information. At least my proposal accounts for the importance of context in deciding what knowledge one needs about an object in order to know what that object is. The proposal is not helpful unless, having identified the task at hand, we can decide what knowledge is needed to do that task. This is the subject of the next section.

One can give a similar account of de re belief reports. If you think, in the de re sense, that John's sister is his wife, you have a belief of the form "n is John's wife", where the name n really denotes John's sister. Let "denotation" name the function that maps a term to its denotation. This function maps the name "Superman" to the man from Krypton, for example. We represent the fact that I think John's sister is his wife by

```
(some n   (believe I ('= n '(wife John)))
        & (denotation n) = (sister John)
)
```

There must be some limitations on the choice of the name n. For example, if Bill is in fact the president of IBM, the name "president of IBM" denotes him. Still believing the tautology "The president of IBM is the President of IBM" will not qualify you as believing that Bill is the president of IBM. The conditions on the name n seem to be weaker in this case than in the "knowing what" examples.

6.5.4 Knowing How

If you know that Mary's number is 5766, you know how to call Mary. When does knowing that P entail knowing how to perform action A? Moore proposed that actions have parameters - for example, the number to be dialed is a parameter of the action of dialing. You know how to do the action if you know what the parameters are. And you know what X is if X denotes one object in all your alternatives. Since "5766" denotes one object in all situations, someone who knows that Mary's number is 5766 knows how to call Mary. Konolige agrees that you know how to perform an action if you know what its parameters are. As mentioned above, Konolige holds that you know what X is if you have a standard name for X. Since "5766" is a standard name, someone who knows that Mary's number is 5766 knows how to call Mary.

Both proposals go wrong in the same way. "Six times thirty-one squared" denotes 5766 in all situations. So if I know that Mary's number is six times thirty-one squared I know how to call Mary according to both Moore and Konolige. And this prediction is wrong. I have to figure out that six times thirty-one squared is 5766 before I can call Mary, and if I don't have pencil and paper handy it may not be easy to call her.

Even if these proposals could be made to work, they are not satisfying. There is no apparent reason why having a standard name should help you to perform an action. A better theory would have more intuitive appeal. It would make us say "Ah, now I see why you need that piece of knowledge to perform that action."

Let us return to our imaginary robot, and ask "How would the robot call Mary on the phone? At which point would he use his knowledge of her phone number?" The robot can act only by executing a program. He knows how to perform an action if he knows what program he should execute to perform that action. Since programs are expressions of the internal language, there is no mystery about when the robot knows what program to execute. He knows what program to execute if he has the quote name of the program. From the quote name he can reconstruct the program itself; he can then proceed to execute it. Our problem is then to show that the robot can construct a program for dialing Mary's number if he knows the arabic numeral for Mary's number.

Intuitively it is obvious why you need to know the arabic numeral for Mary's number to call her. Telephones have arabic numerals printed on their dials. You use those numerals to identify the right holes to put your finger in. If phones had roman numerals printed on them instead, you would need the roman numerals for Mary's number to call her. We must reconcile this common sense observation with the claim that the robot knows how to dial Mary's number if he knows what program to execute in order to dial Mary's number.

The robot performs physical actions by sending commands to his effectors. These are devices that accept commands in the internal language as input, and produce physical actions as output. A command is simply a sentence of the internal language that describes the desired action. If the effectors perform this action the sentence will be true.

Of course the effectors can only accept certain sentences as commands. Even if two sentences describe the same action, it does not follow that if the effectors can accept one sentence as a command they can also accept the other. A real robot can turn one joint of his arm through an angle of n degrees by putting a binary numeral for n in a certain register. No other name for the number n will do.

Actually the commands are not sentences but wffs. A sentence that describes an action must give the time when the action was performed. But when the robot sends a command to his effectors he wants it carried out now; he does not need or want to specify the time. So the robot uses the free variable "Vnow" to stand for the present time in commands to the effectors. If the robot sends a command to his effectors during interval 1, the action will be carried out during interval 1. So the command will be satisfied when the variable "Vnow" is bound to the interval 1. "(CloseHand Robot I)" means that the robot closes his hand during interval 1. The robot uses his selfname to refer to himself in commands to his effectors. So he sends the wff "(CloseHand Me Vnow)" to his effectors when he wants to close his hand. "(command Robot w i)" means that the robot sends wff w to his effectors during interval i. So the robot believes

(all i (command Me '(CloseHand Me Vnow) i) -> (CloseHand Me i))

This sentence says that if the robot commands his hand to close, it will close.

Let us return to the phone dialing problem. Suppose for simplicity that the phone has push buttons rather than a dial. To "dial" Mary's number the robot must tell his hand which buttons to push. The robot's hand is presumably guided by his eye. We assume that the problem of hand-eye coordination is handled by low-level routines that do not concern us. All the robot has to do to direct his hand to a certain object is to supply the coordinates of that object in the visual field. As mentioned above, the visual field is a rectangle, and the robot uses Cartesian coordinates to specify positions in the visual field. "(push Robot x y I)" means that during interval i the robot's hand reaches out and pushes the object at coordinates (x,y) in the robot's field of view. When the robot commands his hand to push the object at coordinates (x,y), he uses arabic numerals to specify the coordinates x and y. So if the robot issues the command

('push 'Me (arabic x) (arabic y) 'Vnow)

his hand will push the object at coordinates (x,y).

When the robot points his eye at the buttons on the phone, he will see the arabic numerals from "0" to "9" printed on the buttons. As stated in Section 6.5.1.2, the sensors will report that a numeral n is written at coordinates (x,y) by producing the sentence

('WrittenAt (quote n) (arabic x) (arabic y) 'Interval99)

The numerals (arabic x) and (arabic y) are precisely the data structures the robot needs to build a command that will cause his hand to push the button with the numeral n printed on it. If the sensors produce the sentence

(WrittenAt '5 128 100 Interval99)

the robot knows that the button with the numeral "5" printed on it is at coordinates (128,100) in his field of view. He can push it by issuing the command

(push Me 128 100 Vnow)

So assume that the robot is looking at the telephone. He can dial the number "5766" by executing a program that looks roughly like this:

```
Scan until you receive a percept of the form
(WrittenAt '5 x1 y1 i);
Send the command (push Me x1 y1 Vnow);
Scan until you receive a percept of the form
(WrittenAt '7 x2 y2 i);
Send the command (push Me x2 y2 Vnow);
Scan until you receive a percept of the form
(WrittenAt '6 x3 y3 i);
Send the command (push Me x3 y3 Vnow);
Scan until you receive a percept of the form
(WrittenAt '6 x4 y4 i);
Send the command (push Me x4 y4 Vnow);
```

Now it is clear why the robot needs the arabic numerals for the digits of Mary's phone number to construct a program for dialing her number. He needs to find the the right buttons to push, and he identifies them by the arabic numerals printed on them. The robot cannot dial the number in a pitch black room. Moore and Konolige treat dialing as a primitive action. They do not mention that you have to look for the buttons with the right numbers printed on them, so they do not predict any difficulty about dialing a telephone in the dark. They can of course assert that having light is a precondition of dialing. But it is better to derive this precondition from the general rule that you can't see in the dark.

6.5.5 Belief and Truth

Common sense says that snow is white if and only if it is true that snow is white. One can formalize this idea with a Truth Schema. For every sentence p, the sentence

('<-> ('true (quote p)) p)

is an instance of the Truth Schema. This schema says that the truth of a sentence depends on the properties of the objects mentioned in the sentence. For example, one instance of the Truth Schema is the sentence

(true ('white 'snow)) <-> (white snow)

which says that the sentence "(white snow)" is true iff snow is white.

Now we can formalize the inferences involving truth in Section 6.2. The following inference is correct:

John believes that gold is an element.
Everything that John believes is true.
____
Gold is an element.

The formal translations of the premisses are:

(believe John ('element 'gold) I)
(all x (believe John x I) -> (true x))

These sentences entail

(true ('element 'gold))

The following is an instance of the truth schemP:

(true ('element 'gold)) <-> (element gold)

The last two sentences entail

(element gold)

that is, gold is an element.

The Truth Schema captures our intuitions about truth nicely. Unfortunately, our intuitions about truth are not consistent. The problem is the celebrated liar paradox. Suppose I say "This statement is false". If the statement is true, it is false; and if it is false, it is true. We can get a formal version of this contradiction by assuming

p = '(~ (true p))

The following is an instance of the Truth Schema:

(true '(~ (true p))) <-> (~ (true p))

Substituting equals gives

(true p) <-> (~ (true p))

which is an obvious contradiction.

How we deal with this problem depends on what we think the goal of Artificial Intelligence is. If we are trying to make machines as intelligent as possible, we must abandon the Truth Schema and look for a new schema that can handle the Liar sentence without contradiction. There are several ways to do this. For example, see [15].

On the other hand, if we are trying to make machines as intelligent as people, we don't want to give them a solution of the Liar Paradox even if we know of one. Ordinary people can't resolve the Liar Paradox; they can only note that it is a paradox, and go on using the Truth Schema as before. If our machines are only supposed to be as intelligent as ordinary people, they should do the same. This does not mean that we should put the Truth Schema into our logic and forget about the matter. If we do that, we have no way of knowing when the contradictions will appear or how much trouble they will cause. Even if we find by experiment that no problem arises in this or that application, we can't just ignore the problem. It is our job, not just to build programs that work, but to understand why they work. Our task is not done until we answer the question "How can machines (or people) get away with using an inconsistent theory of truth?".

Let us look again at the Liar example, p = '(~ (true p)). Suppose we try to discover whether this sentence is true by using the usual Tarskian rules for assigning truth values, along with the rule that (true x) is assigned the same truth value as x. The sentence is the negation of '(true p), so to find its truth value we must find the truth value of '(true p). To find the truth value of this sentence we must find the truth value of p. But p is the sentence we started with. The attempt to find the truth value of p thus leads to an infinite loop. A sentence is called grounded if we can find its truth value by the given rules without infinite loop.

Kripke [9] pointed out that many quite ordinary utterances can be ungrounded if circumstances are very unfavorable. Suppose Joe Smith is walking down a road at noon on July 1, 1982. He sees a sign by the road, too far away to read, and remarks "The statement on that sign is true." He approaches the sign and reads the words "The utterance of Joe Smith at noon on July 1, 1982 is false". If we attempt to find the truth value of this sentence by usual rules we get an infinite recursion. But of course the example was created only by assuming a very peculiar road sign. Although many utterances could lead to this kind of infinite recursion, ir practice not many do.

Following Kripke, one can give a formal definition of this notion of a grounded sentence, and prove that in any model we can choose the extension of the predicate "true" so that (true 'p) <-> p holds for every grounded sentence. Since ungrounded sentences seldom arise in practice, they are rare in the intended model of the robot's beliefs. Therefore most instances of the Truth Schema are true in the intended model. And that is why it is safe for the robot to use the Truth Schema.

6.6 Further Work

Further work on these lines could be of two kinds: improvements in the theory, and applications of the theory. The theory has a major shortcoming as it stands: unlike Moore's theory, it does not include a formal theory of planning. Since my treatment of time uses intervals, not situations, one cannot simply add situation calculus. It would be straightforward to get rid of the intervals and add situation calculus to the theory. But situation calculus has its own problems, and people are working on better treatments of time [16]). It would be nice to keep the interval theory of time and find a planning theory based on intervals rather than situations. This problem is tackled in [12], [1] and [5].

One could make several other extensions to the theory, but real progress will come only from studying applications. A program can use this theory in two ways: to reason about its own beliefs and other people's. Planning programs need to reason about their own beliefs so that they can plan to acquire knowledge, either for its own sake or as a prerequisite to further actions. Since most planning work to date has used situation calculus, one might replace intervals with situations before applying the theory to planning. The theory would then allow a program to build plans involving perception, introspection, inference and physical actions.

Reasoning about other people's beliefs is important in interactive programs (whether or not they use natural language) and in story understanding. With a good representation of belief one can assert (for example) that agent A is lying to agent B, while B realizes that A is lying but pretends to be fooled. Here one would like to use knowledge of an agent's beliefs to predict his actions, a topic considered in [4]. This type of application should provide evidence for a better theory of knowing what.

Some people hope that AI programs in all domains can benefit from knowledge about their own knowledge. One might express heuristics by saying "This piece of knowledge is good for solving this type of problem". One could describe a default by saying "Assume that a human being has two arms unless you have knowledge to the contrary", thus avoiding the pitfalls of non-monotonic logic. These ideas are attractive but untested.

REFERENCES

[1] Allen, James. A General Model of Action and Time. Department of Computer Science, University of Rochester, September, 1981.

[2] Creary, L. G. "Propositional Attitudes: Fregean Representation and Simulative Reasoning." *IJCAI-6* (1979), 176-181.

[3] Fodor, J.A. Three Cheers for Propositional Attitudes. In *Representations*, Bradford Books, Montgomery, Massachusetts, 1982.

[4] Haas, Andrew. *Planning Mental Actions*. Ph.D. Th., Department of Computer Science, University of Rochester, 1982.

[5] Haas, Andrew. What Robots Do and What Robots Can Do. unpublished manuscript

[6] Hintikka, J. Semantics for Propositional Attitudes. In *Reference and Modality*, L. Linksy, Ed.,Oxford University Press, London, 1971, pp. 145-167.

[7] Kaplan, D. Quantifying In. In *Reference and Modality*, L. Linsky, Ed.,Oxford University Press, London, 1971, pp. 112-144.

[8] Konolige, K. A First-Order Formalization of Knowledge and Action for a Multi-Agent Planning System. Tech. Rept. 232, SRI International, 1980.

[9] Kripke, Saul. "Outline of a Theory of Truth." *Journal of Philosophy 72*, 13 (1975).

[10] Lycan, William G. "Towards a Homuncular Theory of Believing." *Cognition and Brain Theory 4*, 2 (1981), 139-157.

[11] McCarthy, John. First-Order Theories of Individual Concepts and Propositions. In *Machine Intelligence 9*, Halsted Press, New York, 1979, pp. 120-147.

[12] McDermott, Drew. A Temporal Logic for Reasoning About Processes and Plans. Tech. Rept. 196, Department of Computer Science, Yale University, March, 1981.

[13] Moore, Robert. Reasoning About Knowledge and Action. Tech. Rept. 191, SRI International, Menlo Park, California, 1980.

[14] Moore, Robert, and Hendrix, Gary. Computational Models of Belief and Semantics of Belief Sentences. Tech. Rept. 187, SRI International, Menlo Park, California, 1979.

[15] Perlis, Donald. *Truth, Syntax and Reason*. Ph.D. Th., Department of Computer Science, University of Rochester, 1980.

[16] Vilain, Marc. A System for Reasoning About Time. Proceeedings of the 1982 National Conference on Artificial Intelligence, AAAI, 1981, pp. 197-201.

7. A TRANSPORTABLE NATURAL LANGUAGE INTERFACE[1]

M. Bates and R.J. Bobrow

7.1 Introduction

This paper describes work in progress to develop a facility for natural language access to a variety of computer databases and database systems. This facility, called IRUS for Information Retrieval using the RUS parsing system, allows users who are unfamiliar with the technical characteristics of the underlying database system to query databases using typed English input. This system can be thought of as a stand-alone query system or as part of a management information system (MIS) or a decision support system (DSS).

Many systems boast of having a "user-friendly" or "English-like" or even "English" interface so that users require a minimum of special training to use the system, but most such systems use shallow, relatively ad hoc techniques that are not robust or linguistically sound. We are using a large, well-tested, theoretically-based, general parser of English that has been developed and extended in a variety of research projects for over a decade.

One of the primary emphases of IRUS is transportability, which includes three types of changes: (1) changing the domain, (2) changing data bases

[1] This paper appears in the proceedings of the 6th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Bethesda, Maryland, June 6-8, 1983.

within the same domain, and (3) changing data base systems. The use of a general parser for English is an important part of the solution to the transportability problem, but there are other parts as well, since portions of the system beyond the parser must know the conceptual content of the domain, the way in which this is reflected in a collection of datasets, and the operating characteristics of the dbms being used to access these datasets.

Other researchers have investigated similar issues [8, 5, 6, 12]. We have attacked this problem by building a knowledge-based system, with procedural components independent of domain and data base structure, directed by domain and database dependent knowledge structures. We are also building tools for conveniently creating and maintaining these knowledge structures, with an eventual goal of allowing end-users to extend and modify these knowledge structures to suit their own needs. Given this set of goals, and these tools, we consider the current implementation, which uses the System 1022 dbms on the DEC KL-2060, to be only one of a set of possible implementations, and are not constraining IRUS on the basis of 1022's strengths and weaknesses.

This paper presents an overview of the IRUS system, emphasizing those aspects of the design that are critical to transportability. We describe the parsing system, which is a completely independent module that has been interfaced to a variety of different applications, and then discuss the other modules which bridge the gap between the parser and the dbms.

7.2 Overview of IRUS

Figure 1 shows a schematic diagram of the IRUS system. The RUS parser forms the linguistic front end, and interacts with an incremental semantic interpreter to produce formal semantic interpretations. These interpretations are represented in a formalism called the meaning representation language, *MRL*, which is discussed in Section 4.

The processing up to this point needs to have knowledge of the particular domain being addressed, but it is completely independent of any particular dbms. Thus, the relevant knowledge bases are the dictionary, which defines the vocabulary to be used in discussing the domain, and a set of semantic interpretation rules which specify the way in which different linguistic constituents are to map to expressions in MRL.

The next step in processing is to fill out the interpretation using additional linguistic (but domain independent) information. This includes:

- resolving pronouns and other anaphoric references,
- filling in ellipsed elements (such as expanding the interpretation of "Is the author from Ireland?" to mean "Is the author of that book on Shakespeare from Ireland?"),
- and disambiguating references by using discourse information (such as identifying which book on Shakespeare was being talked about).

This component is currently the least developed part of the system, as these topics are current research problems.

The resulting interpretation, still expressed in MRL, is ready to be translated into a representation that makes explicit reference to the

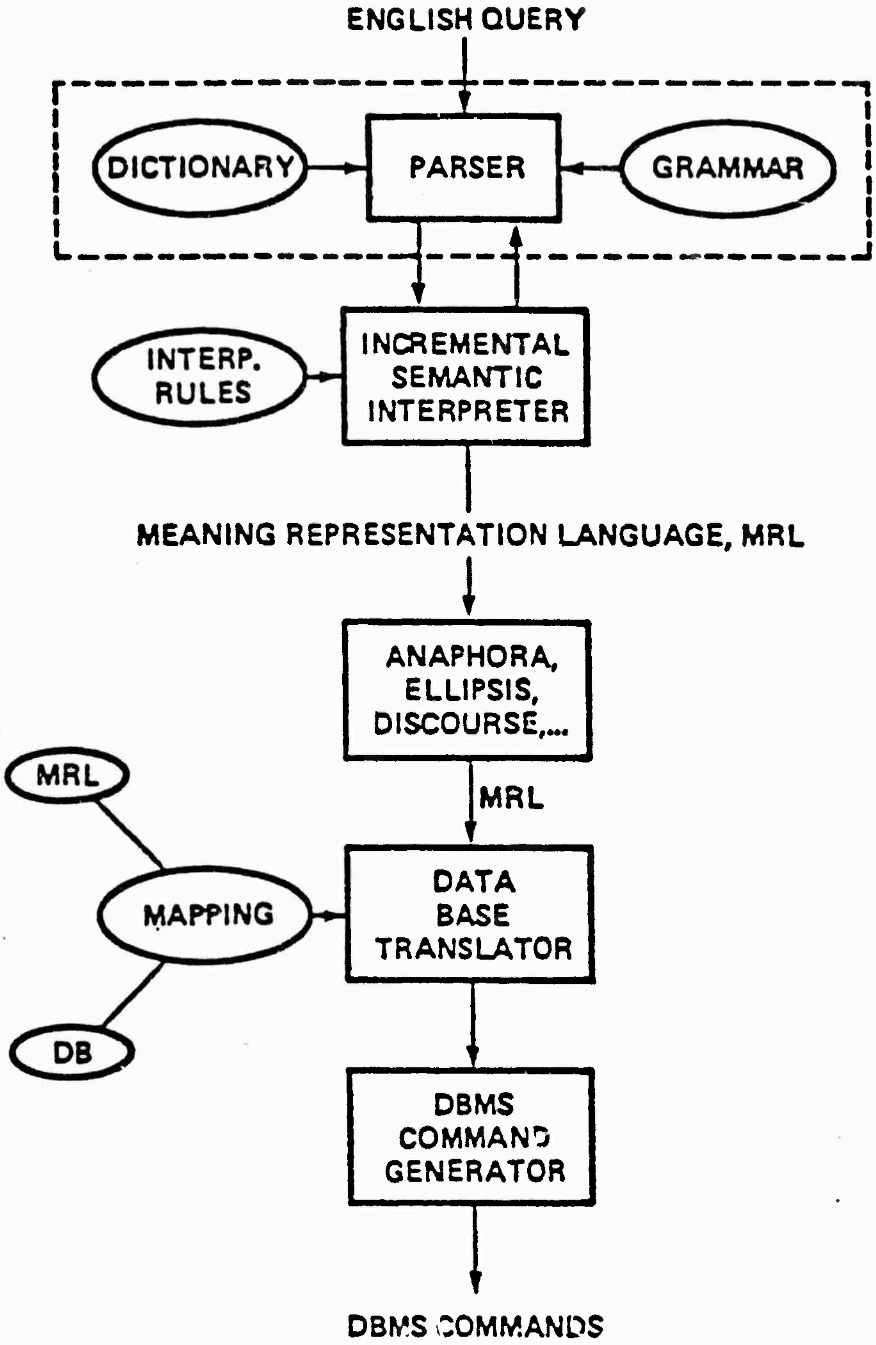


FIG. 1. ORGANIZATION OF THE IRUS SYSTEM

structure of the particular data base being used. The function of the data base translator is to take the MRL interpretation and, using information about the relationship between the real-world facts (the user's conceptual structure) and the particular data base being used (specific file names, record and field names). For example, the fact that a book with a particular Library of Congress number was acquired on a particular date might be represented in a variety of ways in different data bases, such as (1) fields named LC# and ACQDATE within a BOOK record, whose values area number and a date respectively, or (2) an ACQUISITION record with fields name DATE and BOOKID, where BOOKID is an internal identifier for a book that keys into a record BOOK which has a field for the Library of Congress number.

The final stage of processing is to transform this data base representation language into a sequence of interactions or commands in the dbms query language that will produce the answer to the initial query. This involves optimization to reduce search, based on the indexing characteristics of the dbms.

7.3 The RUS System

The RUS parsing system is based on the formalism of Augmented Transition Networks [1, 16]. It is a highly modular system, consisting of a grammar, a parser, a lexical component, and an interface for communication of partial results to and from a separate semantic component. The details of the parsing mechanism will not be given here; they are discussed in [4] and [2](in progress). Instead, we briefly sketch the capabilities and performance characteristics of the parser, stressing the fact that the RUS design expresses syntactic constraints in a broad, general way while using tight semantic constraints (if they are available) to guide the parsing and to interpret the resulting structure.

7.3.1 The Grammar

When operating without semantic or other constraints, the output of the grammar is a syntactic case structure, or labeled tree, in which a variety of syntactic slots (HEAD, NUMBER, SUBJECT-NP, etc.) are filled by atomic entries or other case structures. This case structure represents in some sense a normalized breakdown of the input sentence. It preserves some information about the surface structure of the input (for example, what was the first noun phrase) while at the same time extracting common information from different surface structures. For example, the parses of the two sentences "John borrowed the book" and "The book was borrowed by John" both indicate that "John" is the subject of "borrow" and "book" is the object, but the latter parse contains information that the sentence was passive.

The RUS parser uses an ATN grammar consisting of 92 states and 322 arcs. The syntactic coverage of the RUS grammar is quite broad, larger than the LUNAR system [17], and comparable to the SRI DIAMOND system [7] and the LSP parser of Sager, et. al. at NYU [9]. The following list exemplifies some of the structures that can be parsed:

- Which of the reports received from NLM in the last three years concerned treatments for infectious hepatitis?
- What is the average length of the five most widely circulated reports written by our division staff?

- Which organizations that we receive reports from have responded to either of our recent questionnaires?
- Of the books on Artificial Intelligence, how many have been classified as text books?
- Have there been as many requests for books about medicine this year as we planned for in our budget?
- Has anyone from any college or university inquired whether we maintain a distribution list?
- How much money have we spent on books on Artificial Intelligence this quarter?

7.3.2 The Parser

Although the parser can be used without semantic support, it has been designed to be part of a cascaded system [19, 10, 3] in which feedback from other components of a language understanding system (such as semantics, pragmatics, and a dialogue expert) can be used to improve the performance of the parser. The production of the semantic interpretation is interleaved with the parsing process, and the results of the interpretation process are used to guide the parser in producing only semantically acceptable parses.

When operating without semantic or other constraints, the output of the RUS system is a syntactic case structure, or labeled tree, in which a variety of syntactic slots (HEAD, NUMBER, SUBJECT-NP, etc.) are filled by atomic entries or other case structures. This case structure represents in some sense a normalized breakdown of the input sentence. It preserves some information about the surface structure of the input (for example, what was the first noun phrase) while at the same time extracting common information from different surface structures. For example, the parses of the two sentences "John borrowed the book" and "The book was borrowed by John" both indicate that "John" is the subject of "borrow" and "book" is the object, but the latter parse contains information that the sentence was passive.

The parsing system is written in InterLisp and runs on several computers including the DEC Systems 10 and 20, the VAX, the Xerox 1100 Series, and BBN's Jericho personal computer. We expect that it will soon be available in either Franz Lisp or PSL to run on the VAX and 68000 computers. The sample sentences listed at the end of the previous section have an average of 15.7 words and parse in an average of .7 seconds of CPU time using the DEC 2060 version of RUS. This is easily fast enough to be part of a front-end for a system that must satisfy an interactive user.

7.4 The Database Interface

In order to have an appropriately modular system, it is necessary to separate the user's conceptual view of the domain from the details of both the database structure and the particular database system being used. The result of parsing and semantic interpretation should be expressed in terms of the user's conceptual structure so that the database structure can be changed without having to modify the language understanding part of the system. This representation (interpretation) should have a clear formal semantics, so that it is possible to determine whether the system has adequately represented ("understood") the user's input.

The primary semantic representation or MRL used by IRUS is a descendant of the typed quantification language used in the LUNAR natural language datebase system. This MRL has a formal declarative semantics that can be expressed in the predicate calculus as well as a formal procedural semantics [18, 20]. It is an extended form of the predicate calculus in which the domains of quantification for variables are made explicit. The general form is:

```
(FOR <quant> X / <class> : (p X); (q X))
```

- o where
- o <quant> is a specific quantifier such as EVERY, SOME, THREE, HALF, etc.,
- o X is the variable of quantification,
- o <class> is the domain of quantification (the set over which X can range), such as PERSON, DEPARTMENT, BOOK, MONTH, etc.,
- o (p X) is a predicate that restricts the domain of quantification (such as (IN-DEPARTMENT X DEPT45)),
- o (q X) is the expression being quantified (either a predicate such as (MALE X) or an action such as (PRINT X)).

Consider the sample input sentence "Show the books charged out by people in department 45 in January." The corresponding MRL expression, generated by the semantic interpreter, has the form:

```
(for all x / book:
  (for some y / person:
     (for the z / dept:
          (= (dept# z) 45);
          (dept-member y z));
     (for some w / day:
          (in-month w January);
          (borrow x y w)));
   (print x))
```

There are a number of requirements for a MRL, the most important of which, according to [18], are:

o It must be capable of representing precisely, formally, and unambiguously any interpretation that a human reader can place on a sentence.

o It should facilitate an algorithmic translation from English sentences into their corresponding semantic representations.

o It should facilitate subsequent intelligent processing of the resulting interpretation.

The procedural semantics for MRL makes it possible to define the operation of the system in any database which can provide (1) generators for the entities that are referred to by noun phrases, and (2) procedures to filter such generators to represent predicates. Any database system for which such a procedural semantics can be defined can be interfaced to the MRL output. From our experience, this includes the relational data base systems, and we believe it extends to CODASYL networks as well.

To provide a concrete system on which to develop our ideas, we chose System 1022, a commercially available relational DBMS which was already in use within BBN. To provide a test of generality, we deliberately did not compare a number of dbms systems to choose the one most amenable to this application; we simply used the first dbms that was available to us.

7.5 Conclusions and Future Directions

The initial version of IRUS was operational at the end of January 1983, and we expect to use it as a basis for a number of experiments and practical in-house applications. The existence of a mature, well-engineered parser, the power of the InterLisp program development environment, as well as our past experience on earlier natural language query systems like LUNAR have made it possible to implement IRUS with only a few person-months of effort. Our experience in working with several systems based on RUS leads us to expect that the resulting system will be readily modifiable, as well as transportable.

One of our next tasks will be to implement a user interface that will make possible the semi-automatic construction of domain and dataset models, to make it possible to transport IRUS to new domains using the 1022 system. In order to do this, we will define a formal specification language for domain and dataset models, which are now specified only as abstract data structures, and tie this in with a simple inference mechanism and a flexible human interface.

In the long term, a number of important research problems still exist in such areas of anaphora resolution, ellipsis processing and the discourse model. The initial IRUS system will include partial solutions to these problems which have been developed in previous natural language research projects at BBN, the University of Pennsylvania and SRI. Because the RUS system is also being used in several ongoing research projects which are focussing on discourse level language problems, we expect that results in these areas will be readily transferable to IRUS.

The modularity of the parser's design, together with the potential for effective guidance from the cascaded interaction with semantic knowledge, will permit us to continue the development of linguistically well-founded extensions to the parser. In particular, we plan to investigate the techniques proposed by Weischedel and Sondheimer [13, 14, 11, 15] to process input that is syntactically ill-formed. They have already started to use the RUS system in their work, and have proposed a number of extensions and modifications to RUS to facilitate the handling of ill-formed input. This would allow IRUS to accept and correctly process queries with minor errors and deviations from "standard" English.

REFERENCES

[1] Bates, Madeleine. The Theory and Practice of Augmented Transition Network Grammars. In Bolc, Leonard, Ed., *Natural Language Communication with Computers*, Springer-Verlag, 1978.

[2] Bobrow, Robert J. and Bates, Madeleine. The RUS System for Parsing and Semantic Interpretation. in preparation

[3] Bobrow, Robert J. and Webber, Bonnie L. PSI-KLONE: Parsing and Semantic Interpretation in the BBN Natural Language Understanding System. CSCSI/CSEIO Annual Conference Proceedings, 1980.

[4] Bobrow, R.J. The RUS System. Tech. Rept. 3878, BBN, 1978. (section of the annual report)

[5] Boguraev, B.K. and Sparck Jones, K. How to Drive a Database Front End Using General Semantic Information. Proceedings of the Conference on Applied Natural Language Processing, ACL and NRL, Santa Monica, California, February, 1983.

[6] Ginsparg, Jerrold M. A Robust Portable Natural Language Data Base Interface. Proceedings of the Conference on Applied Natural Language Processing, ACL and NRL, Santa Monica, California, February, 1983.

[7] Grosz, B. et.al. TEAM: A Transportable Natural-Language System. Tech. Rept. No. 263, SRI Artificial Intelligence Center, April, 1982.

[8] Grosz, Barbara J. TEAM, a Transportable Natural Language Interface System. Proceedings of the Conference on Applied Natural Language Processing, ACL and NRL, Santa Monica, California, February, 1983.

[9] Sager, Naomi. *Natural Language Information Processing: A Computer Grammar of English and Its Applications*. Addison-Wesley, Reading, MA., 1981.

[10] Sidner, C.L.; Bates, M.; Bobrow, R.J.; Brachman, R.J.; Cohen, P.R.; Israel, D.J.; Webber, B.L.; and Woods, W.A. Research in Knowledge Representation for Natural Language Understanding: Annual Report. Tech. Rept. BBN Report No. 4785, Bolt Beranek and Newman Inc., November, 1981.

[11] Sondheimer, N.K., and Weischedel, R.M. A Rule-Based Approach to Ill-Formed Input. Proc. 8th Int'l Conf. on Computational Linguistics, Tokyo, Japan, October, 1980, pp. 46-54.

[12] Thompson, Bozena H. and Thompson, Frederick B. Introducing ASK, a Simple Knowledgeable System. Proceedings of the Conference on Applied Natural Language Processing, ACL and NRL, Santa Monica, California, February, 1983.

[13] Weischedel, Ralph M., and Black, John. Responding to Potentially Unparseable Sentences. Tech. Rept. Technical Report No. 79/3, Department of Computer and Information Sciences, University of Delaware, February, 1979.

[14] Weischedel, Ralph M., and Black, John E. "If The Parser Fails." *Proceedings of the 18th Annual Meeting of the ACL* (June 1980).

[15] Weischedel, Ralph, and Sondheimer, Norman. A Framework for Processing Ill-Formed Input. Department of Computer and Information Sciences, University of Delaware, October, 1981.

[16] Woods, W.A. "Transition Network Grammars for Natural Language Analysis." *Communications of the ACM* (13 1970), 591-606.

[17] Woods, W.A. et.al. The Lunar Sciences Natural Language Information System: Final Report. Tech. Rept. 2378, Bolt Beranek and Newman Inc., 1972. (Available from NTIS as publication N72-28984)

[18] Woods, W.A. Semantics and Quantification in Natural Language Question Answering. In M. Yovits, Ed., *Advances in Computers*, Academic Press, 1978, pp. 1-87.

[19] Woods, W.A. "Cascaded ATN Grammars." *American Journal of Computational Linguistics* 6, 1 (January-March 1980).

[20] Woods, W.A. Procedural Semantics as a Theory of Meaning. In A. Joshi, B.L. Webber and I. Sag, Ed., *Elements of Discourse Understanding*, Cambridge University Press, 1981.

8. PROGRESS REPORT: THE RUS SYSTEM

M. Bates and R. Ingria

8.1 Introduction

This year has seen continued extension and improvement of the RUS parsing system, resulting in a more robust, easier-to-use, more complete parsing tool.

Many of the changes in the RUS grammar and dictionary feature system were the result of research reported in [4]. This research consisted of two parts: (1) a comparison of the constructions covered by the RUS parsing system with those handled in other parsing systems, as reported in such works as [7] and [11]; and (2) an examination of the various types of complements taken by verbs, adjectives, and nouns, as listed in such reference works as [10]. All the complement constructions dealt with in the sources consulted are catalogued in [4] and each construction is cross-referenced by the name which it is given in individual sources. This allows for inter-translation among the various grammatical and computational studies consulted.

8.2 The Dictionary

The dictionary has been increased by about a thousand words to its current total of 4700 base and irregular forms. Most prepositions, irregular verbs, and irregular nouns are now in the RUS dictionary.

All forms of irregular verbs have been entered which are in use in current American English. For example, verbs such as "bend, bent" and "sing, sang, sung" have been entered, while archaic or obsolete forms, such as "sod" as the past tense of "seethe", and British English forms, such as "dipt" as the past tense of "dip", have been deliberately omitted. Also, in the case of verbs which have both regular and irregular forms (for example "lean" with past forms "leaned" and "leant"), the irregular forms of these verbs were also entered, so that either past tense can be used.

To insure that the dictionary include all irregular verbs and nouns currently in use, a search was made of several reference works, including [2, 3], [5] and [10], and our word list was checked against a recent dictionary [9]. In addition, where the sources consulted listed a verb which is now obsolete, but which possessed a past participle that survives in an adjectival use such as "accursed" and "molten", these forms were entered as adjectives.

All irregular nouns with unpredictable plurals, such as "child, children" and "goose, geese" have been entered, as have the most frequently used compounds containing them, such as "fireman, firemen". In addition, the most common nouns with foreign plurals have been added, for example Latinate nouns ending in "-um, -a", such as "erratum, errata", Greek nouns ending in "-ex, -ices", such as "vertex, vertices", etc. (For a full list of the classes of nouns entered, see Sections 4.47-4.57 of [10]. See also [8], Chapter XXV.) To insure the most complete coverege of these classes of nouns, Walker's "backwards" dictionary [12], which groups words by endings, was consulted.

This search revealed that there are many highly technical nouns in some of these categories, such as "endothelium", which it would not be appropriate to enter in the RUS dictionary. Only those forms appearing in [9] were entered. In the case of those nouns with regular and irregular plurals, such as "index, indexes/indices", both plural forms were entered. Fin^lly, nouns which are irregular by virtue of the fact tnat they appear only in the plural were also entered. These include historically irregular plurals with no true corresponding singular form at present, such as "kine", and "pluralia tantum" nouns, i.e. nouns which are plural in form but singular in meaning, such as "pants". (See [10], Sections 4.33 and 4.34 and [8], Chapter XXV.)

All single word prepositions listed in [1, 3] which are in current use, such as "aboard" and "underneath", have been entered, but obsolete forms, such as "adrwn" and "overthwart" have not. (Once again, [9] was used to determine the currency of a form.) In addition, many multi-word prepositions, such as "alongside of" and "on top of", listed in [1] were entered. Only those multi-word prepositions which are idiosyncratic in form were entered; those which are more productive were dealt with in other ways. (For example, [1] lists 14 multi-word prepositions beginning with "from". Rather than enter all 14 separately, the dictionary entry to "from" contains a feature indicating that it may take a prepositional phrase as an object to handle these cases.)

The first few hundred most frequently used words of English (according to [6]) have also been added to the dictionary.

8.3 Lexical Acquisition

The lexical acquisition component has been modified in accordance with the new feature set being used. A help facility is being added which allows a user unfamiliar with grammatical terms to get detailed explanations, examples, and negative examples of all the properties asked about in the lexical acquisition dialogue.

When a sentence in mixed upper and lower case is given to the parser, and one of the capitalized words in the sentence is unknown to the lexical component, the word is assumed to be a proper noun (the user is asked to confirm this, but does not have to do anything further to define the word).

A facility has been created to allow special syntactic classes such as dates (4/12/83), times (4:55, 9AM), social security numbers (444-56-7777), monetary expressions ($12.99, $50K) and phone numbers (491--1850x3634) to be recognized in the prepass phase of the parser. These are entered into the chart as single entities. The mechanism for defining special classes is extensible, that is, given a recognition function which takes an atom as its argument and returns a CHARTDEF structure for the atom, it is trivial to integrate that function into the prepass that builds a chart for the parser. This facility could be used to sensitize the RUS system to domain-specific words or numbers such as ID numbers, part numbers, report numbers, job numbers, etc.

8.4 Documentation of RUS

A draft of "The RUS Parser User's Guide" has been prepared, partially funded by the National Library of Medicine. Current users of the RUS system (ISI, the National Library of Medicine, and GTE Laboratories) are providing useful feedback on the usability, documentation, coverage, and other aspects of the system; a final version of the User's Guide will appear when we have assimilated that feedback.

REFERENCES

[1] Curme, George O.. *A Grammar of the English Language, Volume 1: Syntax*. D.C. Heath and Company, Boston, 1931.

[2] Curme, George O.. *A Grammar of the English Language, Volume 2: Parts of Speech and Accidence*. D.C. Heath and Company, Boston, 1935.

[3] Curme, George O.. *English Grammar*. Barnes & Noble, Inc., New York, 1947.

[4] Ingria, Robert J. Syntactic Categories and Subcategories in English. BBN, to appear.

[5] Jespersen, Otto. *A Modern English Grammar on Historical Principles, Part VI: Morphology*. George Allen and Unwin, Ltd., London, 1942.

[6] Kucera, Henry and Francis, W. Nelson. *Computational Analysis of Present-Day American English*. rown University Press, Providence, Rhode Island, 1967.

[7] O'Malley, Michael H. The Children of Lunar: an Exercise in Comparative Grammar. Tech. Rept. ERL-543, SUR Note 189, Computational Speech and Language Group, Electronics Research Laboratory, University of California at Berkeley, 1975.

[8] Poutsma, H.. *A Grammar of Late Modern English, Section I,A: Nouns, Adjectives and Articles*. P. Noordhoff, Groningen, 1914.

[9] Procter, Paul et.al., eds.. *Longman Dictionary of Contemporary English*. Longman Group Ltd., Harlow and London, 1978.

[10] Quirk, Randolph and Greenbaum, Sidney. *A Concise Grammar of Contemporary English*. Harcourt Brace Jovanovich, Inc., New York, 1973.

[11] Sager, Naomi. *Natural Language Information Processing: A Computer Grammar of English and Its Applications*. Addison-Wesley, Reading, MA., 1981.

[12] Walker, J.. *The Rhyming Dictionary of the English Language (Revised and Enlarged)*. Routledge and Kegan Paul, London and Henley, 1924.

9. THE PRAGMATICS OF NON-ANAPHORIC NOUN PHRASES[1]

C.L. Sidner

9.1 Introduction

This paper[2] is an exploration of some issues in the understanding of non-anaphoric definite noun phrases (hereafter defnps) and of those indefinites beginning with "a." In particular I examine what kinds of knowledge a hearer can use to understand such noun phrases, as well as what constitutes that understanding. I will emphasize the discourse and pragmatic components of understanding, that is, knowledge of the discourse context and the speaker's goals and beliefs; I will occasionally point out how various syntactic and semantic features of these noun phrases can be combined with such knowledge to allow the hearer to decide the way the noun phrase was used.

I approach the problem of the the kinds of knowledge the hearer uses in noun phrase understanding from a process orientation. That orientation affects my research in two ways. First, throughout this paper I will be considering language use on the basis of the interaction of a speaker and

[1] Material in this paper was presented in an Invited Address to the Fifth Annual Conference of the Cognitive Science Society, Rochester, NY, May, 1983, under the title "A Computational View of the Pragmatics of Noun Phrases".

[2] This paper has evolved from fruitful discussions with Barbara Grosz on all sorts of aspects of all sorts of noun phrases. In addition, I have benefited from discussions with Rusty Bobrow, Brad Goodman, David Israel and Marc Vilain on drafts of this paper.

hearer; I will not treat language as though it exists independently from the occasions in which it is used. While the "independence" approach is very fruitful for some language research, I find it necessary to consider the process of communication between speaker and hearer. Second, the process orientation affects the kinds of questions I ask about hearer's knowledge, in particular: what role does each kind of knowledge play in the understanding of the hearer, how do those sources of knowledge interact, and what limits the hearer's understanding when one or more kinds of knowledge are not available? In this paper I want to consider just what those sources of information are.

An exploration of the sources of knowledge a hearer uses cannot be undertaken without also considering what the hearer is understanding. I take understanding to mean that a hearer can characterize a noun phrase as having one or more characteristics. The hearer uses these to decide how the noun phrase was used: as a description (and what it describes) or as a referential phrase (and what the referent is) or as underspecified in exact use. This description of understanding depends crucially on understanding what characteristics a noun phrase has in order to judge its use.

Within the class of defnps and a-indefinites there are many distinctions to be drawn about a noun phrase's characteristics, both in terms of intended use and internal structure, and these distinctions affect the way in which a hearer interprets such noun phrases. Previous characterizations of these noun phrases have focused on such distinctions, as for example, whether the noun phrase is intended to be interpreted referentially or attributively. I first review the major characterizations that have been proposed. Later I will draw on these characterizations of noun phrases. Before I can do so, I must introduce a different framework which clarifies how the hearer is able to use his/her knowledge to characterize a noun phrase and understand its use.

Within the process oriented framework that dominates this paper, I will concentrate on what knowledge is brought to bear and what results from using that knowledge. Often I will try to tease out the hearer's decision with an account that includes a model of processing, that is, an account of how the understanding takes place. In this paper I make no claims for the psychological or computational validity of such models. Rather my methodology assumes that processing is never totally divorced from knowledge; first one must be clear about the decision a process computes and then one can detail how the decision is realized. Nevertheless, it is sometimes difficult to imagine that a certain kind of knowledge _is_ used without providing at least one plausible picture of how it _gets_ used, so from time to time I will suggest what processes could be at work.

Before I turn to the main body of the paper, I want to describe in more detail what a hearer must determine about the speaker's intended use of a noun phrase. A hearer must decide whether to take a noun phrase as picking out a individual or rather as providing a description; in the latter case the speaker may have a referent in mind for the noun phrase, but the hearer is not expected to, due to a lack of knowledge about some aspect of the phrase or referent-individual. Consider the following sentence.

s1 _The first place winner of the Boston Marathon_ got a cash prize.

In s1 the defnp may be used simply as a description the speaker thought appropriate, or as a referential description that would tell the hearer just who got a prize, or as a description for which the speaker actually knows the referent but the hearer knows simply that some person so described exists. To understand the noun phrase the hearer must determine just which way the phrase was used.

Hearers may be intended to identify a referent. In such cases, not only does the speaker think the hearer can determine the referent, but the speaker wants the hearer to do so in the course of understanding the discourse. The speaker may believe that the identification is possible either because the defnp is sufficient to distinguish someone (or some thing) already in the hearer's mind or because the hearer can use the defnp to locate the proper someone (the referent) in the world. In D1-1 the hearer is called on to identify the referent; whether the hearer already has a mental entity to which the defnp corresponds or s/he is actually to locate some appropriate person cannot be discerned without further information about the speaker and hearer of this discourse.

D1-1 I saw the handsome man I met at Clarence's again today.
2 Since you know him, could you call him and ask him to dinner?

It is appropriate to combine a discussion of defnps with one of a-indefinites because the two types of noun phrases may be used in similar ways. Compare the alternate second sentences in the sample below:

D2-1 While I was at MIT, I went to the AI lab.
2 a) I'd like you to meet the woman professor I met there.
b) I'd like you to meet a woman professor I met there.

The two sentences seem quite similar as in both the speaker is talking about a particular woman professor met at MIT and one that the hearer is assumed not to be familiar with. Also, in D2-2a, the speaker does not assume that the hearer knows the referent of the defnp (in contrast to D1-1). Since both the defnp and the a-indefinite may be used in other ways, we can ask what knowledge the hearer uses to disambiguate the intended use of the defnp from other uses and to recognize the a-indefinites use as signifying one particular individual. We may also ask whether the two uses in D2 communicate some different information.

A-indefinites can be used in other ways as well.

> s2 A new employee will be serving on the labor relations committee.

In s2, the speaker is either saying:

1. that there is at least one new employee who will participate in the committee,
2. or of some particular new employee that s/he will participate in the committee.

The ambiguity centers on whether the speaker has a particular object in mind to discuss or instead knows that at least one object is correctly described by the a-indefinite and wants to discuss such an object. This paper will explore just what knowledge the hearer can use to determine which interpretation was meant.

In order to explore the issues I have raised, I will first categorize a number of characteristics that have been associated with defnps and a-indefinites by others. I will also show that a pragmatic theory of the hearer's interpretation of noun phrases must postulate and relate several distinct ways in which a hearer is intended to understand defnps and a-indefinites.

The second part of this paper presents a framework for integrating the knowledge which the hearer brings to bear in characterizing a noun phrase. In particular, I will investigate why the hearer uses the following types of knowledge to determine these characteristics, and why they in turn allow the hearer to choose the pragmatic interpretation that the speaker intended:

- the speaker's intention regarding the actual communication given to the hearer,
- the speaker's overall goals, both communicative and otherwise,
- the beliefs of the speaker and hearer,
- the role of focusing ([10], [11], [17], and [18])
- the effect of Grice's [9] maxims of conversation.

9.2 Setting the Stage: Previous views on defnps and a-indefinites

Barwise and Perry [1] outline a view of defnps in which they formalize situation types (which are reflected in statements in English) as partial functions from relations and sequences to T and F. A defnp is said to be *value free* relative to a set of situations when the elements of the phrase are interpreted without reference to a single distinguished situation. Thus the defnp in "the first president of the US had to be less than 6 feet tall" is value free if it is interpreted simply by combining the semantics of its component parts. When the defnp is given a value in a particular situation (e.g. Washington as the first president of the US), the use is *value loaded*. Normally speakers and hearers jump to value loaded interpretations when they know a distinguished context, especially the context of the real world. Sometimes, however, the speaker can intend that the value free interpretation be used instead.

Not only does Barwise and Perry's distinction capture the P‘ssellian and Strawsonian view of reference in one framework, but, more important for the current purpose, it expresses in semantic terms a natural distinction in the use of language, the distinction between descript:on and reference. It thus can lead to a valuable fanning out of pragmatic distinctions in the uses of defnps. Just which uses are characterized will be discussed later.

The notions of value free/loaded defnps are closely associated with Donnellan's [5], [6] attributive and referential distinction. Donnellan exemplified the attributive use of a defnp by the following situation. Two people come upon a friend Smith who is lying on the ground and is dead of foul wounds. One of them says "The murderer of Smith must be insane." Neither person knows who the murderer is, and in fact there may be no murderer, but the defnp use is legitimate all the same, because, as Donnellan claims, the speaker is not really referring to anyone in particular, but rather characterizing whoever, if anyone, falls under the description. Barwise and Perry see value free/loaded interpretations as corresponding to Donnellan's attributive and referential uses respectively. The reason is that in both the value free and attributive cases, the speaker is not trying to refer to an entity, but rather is using the description to attribute properties to any entity that satisfies the definite description. In the value loaded and referential cases, the speaker has a specific situation and individual in mind and is pointing to that individual.

Defnps can be characterized with a different semantic attribute, that of describing or referring to a specific object (the so-called specific use) or of mentioning a generic class (the generic use). The two cases are exemplified below:

s3 Take the elephant for a walk, please.

s4 The elephant is a large and powerful beast that dwells in jungle terrain.

A-indefinites also share this dual characterization as specific or generic:

s5 Take an elephant for a walk today. You'll be glad you did.

s6 An elephant is a large and powerful beast that dwells in jungle terrain.

As Lyons [12] points out, the generic uses of defnps and a-indefinites are not interchangeable since s7 is acceptable English as a generic proposition, but s8 is not.

s7 The lion is no longer to be seen roaming the hills of Scotland.

s8 A lion is no longer to be seen roaming the hills of Scotland.

Some linguists (Lyons and Dahl [4]) seem to think that generic references are characterized by occurring in generic propositions. Sidner [17], however, showed that whether or not a sentence is generic may depend on the preceding context. This leaves open the question of whether hearers first decide the sentence is generic and interpret all the noun phrases accordingly, or whether the hearer uses parts of the sentence to help determine that the whole sentence is generic.

Whether an a-indefinite is interpreted as generic is related to a distinction in the use of specific a-indefinites as either describing a particular object, or as mentioning an object that is representative of one or more objects (e.g. the two interpretations of s2 given previously). Generic a-indefinites usually carry a sense that the object being described is a prototype of a generic class, as in:

sS An elephant can live to be 90 years old.

This prototypical sense is similar to the representative sense of the a-indefinite. While the hearer cannot distinguish any of the uses of a-indefinites with just syntactic or semantic knowledge, if the hearer could eliminate the particular object reading, then it might be easy to distinguish generics from representative on the basis of knowledge about whether or not a prototype reading can be used in the sentence. For researchers interested in organizing memory and inference machinery for language understanding, a required capability of the machine becomes apparent: it must be able to express the distinction between a generic class and prototypes of it and reason about the appropriateness of prototypes in certain semantic relations.

Now let us consider some pragmatic attributes of defnps. Perrault and Cohen [14] point out some difficulties in stating necessary and sufficient conditions for a speaker's referring to an entity x by uttering expression E. They show that the speaker and the hearer must mutually believe that the referring expression E is fulfilled by the entity in the given context. They state certain conditions on mutual belief and then point out that these do not hold for uses where the speaker is seeking information (as in "What is the departure time of the next train to Montreal") or for Donnellan's attributive case (where the hearer need have no beliefs about the existence of the attributed object).

While Perrault and Cohen give conclusive evidence that mutual belief sometimes plays a role, there are other circumstances where it is possible for a speaker to use a definite expression E for entity x when the hearer does not share the speaker's knowledge of x or how it can be referred to. For example, the speaker may say s10, without assuming the hearer knows that there is someone in charge of admissions.

s10 Go to the Alfred Building and see the woman in charge of admissions.

In such cases the defnp acts as a description that the hearer is to use, either to identify the proper referent or simply to permit the speaker and hearer to share additional information in a conversation (e.g., s11)

s11 You know, I still have the ballet costume I wore in my first grade pageant).

Cohen [3] has been exploring types of situations in which the speaker expresses his/her intention for the hearer to identify the object referred to by a defnp. The intention may be stated either indirectly or directly, and is sometimes isolated in a separate sentence. Thus in s10, the speaker intends for the hearer to identify a particular woman as being the woman in charge of admissions, in order to carry out some business with her. Sometimes the requests to identify must be immediately followed by the identification act before more is said, as in Cohen's example below. In this example, a conversation about constructing water pumps, D3-1 is a request to identify.

D3-1 Expert: See the clear elbow tube?
2 Apprentice: Yes.
3 Expert: Place the large end over that same place.

The hearer must identify the tube immediately after the identification request before proceeding to the next step of the construction. Requests to identify are a significant source of knowledge about how a noun phrase is used because they provide conclusive evidence for the phrase's intended interpretation.

I must briefly note an additional class of defnps which are different from those discussed in this paper, that is, those defnps that are anaphoric and those that are in implicit focus. Examples are given below:

D4-1 Jeff owns **a shiny new car**. (focus is in boldface)
2 The engine is so clean you could eat off of it. (defnp in implicit focus)

D5-1 Sally took **her pet snake and hamster** to the park. (focus is in boldface)
2 The hamster was afraid to leave its cage,
3 but the snake slithered away and bit an old man.

I assume, following the work of Grosz [10] and Sidner [17], that such defnps can be easily distinguished due to rules governing anaphoric relations between defnps and the discourse focus, as well as rules governing global focusing and objects in implicit focus. These rules can easily predict when a defnp was used anaphorically or with an implicit focus, and hearers make use of the knowledge of which these rules are an encoding. Given such rules, I assume that anaphoric defnps and implicitly focused objects can be discerned as such, and that no consideration of them as non-anaphoric defnps occurs.

To summarize, the characterizations of defnps and a-indefinites I have described are:

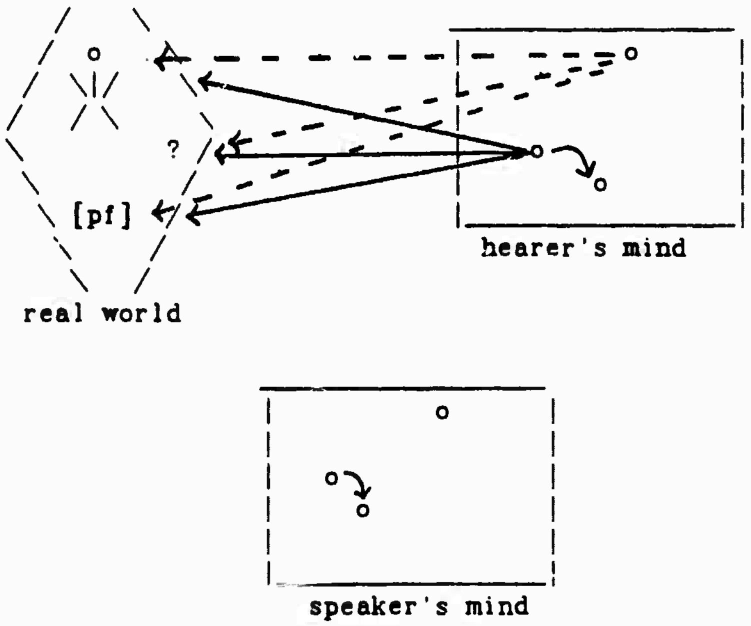
- for defnps: value-loaded/value-free interpretations (Barwise and Perry)
- for defnps: referential/attributive distinction (Donnellan)
- for both: specific/generic noun phrases
- for defnps: mutual belief/speaker belief/no belief in existence of the referent (Cohen and Perrault)
- for a-indefinites: uniqueness of the object referred to/ choice of a representative object for a noun phrase

- o for defnps: speaker intention for hearer to identify (Cohen)

9.3 Distinctions in the use of defnps and a-indefinites

In this section I want to sketch the full array of <u>pragmatic uses</u> of defnps and a-indefinites that are implicit in the preceding discussion. To begin exploring these uses, I will rely on the diagram below to represent the basic mental situation that a speaker or a hearer have when using noun phrases.

FIG. 1. MENTAL STATE OF SPEAKER OR HEARER



The figure 1 may be considered either from the hearer's viewpoint or the speaker's viewpoint. From the hearer's view, the hearer's mind is a collection of descriptions of entities, the speaker's mind is just another part of its mind that holds a collection of entities which the hearer attributes to the speaker (the details of any one entity may differ in the two spaces), and the real world is those actual items in the world.

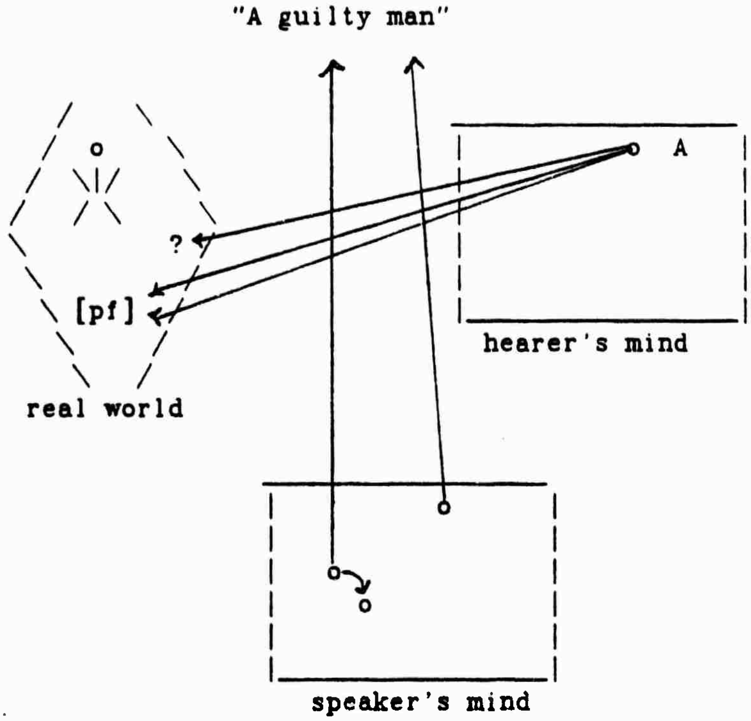
The hearer must always determine how to map entities in its mind onto items in the world. To do so, the hearer makes use of information in the entity as an argument to a mapping function. Sometimes the hearer can map directly to a particular item or individual from the mental entity (represented by the stick figure); sometimes the map only tells him/her that some item or individual is chosen but not which one in particular (represented by a question mark in the figure); this mapping will be called the anonymous individual mapping. Sometimes the map simply indicates that a item will be chosen from a set of possible items without stipulating which one; in Barwise and Perry's terminology the mapping is a partial function, and hence the mapping result is indicated by "pf."

The items in the hearer's mind either have a connection to some previous items[1], (represented by an arrow between two points) or are connectionless. Connectionless items occur when new information, unrelated to any previous items, are introduced into memory. If a speaker's view is taken on figure 1, the hearer's mind is taken as those items attributed to the hearer. In this way, the figure can be used either to consider the speaker's situation in generating noun phrases or the hearer's in interpreting them.

[1] They may specify some previous item in memory. A specification is just an item in a hearer's mind that stands in some representation relation to objects of the world. Specifications are assumed to represent objects in the real world when such objects exist, or to represent the mental schema of properties associated with non-existent objects, such as Santa Claus. See Sidner [17, 18] for details.

How are items introduced into the hearer's mind? One way is by hearing and interpreting noun phrases. Let us first consider the kind of mappings that result from a-indefinites as shown in figure2.

FIG. 2. MAPPINGS FOR "A GUILTY MAN"



There are two different types of mappings to consider: what the noun phrase maps into in the hearer's mind (or correspondingly, the noun phrase mapped from a mental item of a speaker), and what mapping exists between a mental item and the real world.

An a-indefinite maps into a new item in the hearer's mind; we can imagine the new item has two parts: a description of the syntactic and semantic features of the phrase from a sentence (e.g. the head of the phrase is "man", it has a particular semantic role in a sentence, etc.), and a description of the kind of object described by such syntactic and semantic features (a man who has committed some crime). A new item is new because it does not specify any previous items. A-indefinites, by their very nature, can only map to new items for the hearer. However, for the speaker, "a guilty man" could be generated because the speaker has either some previous item to speak about or a new item. On hearing the phrase "a guilty man," the hearer may believe the speaker has in mind some item with many previous connections, but the hearer cannot discern about the speaker's situation how that item maps into the world for the speaker. The hearer does have to determine which mapping into the real world the speaker intended the hearer to use for the a-indefinite.

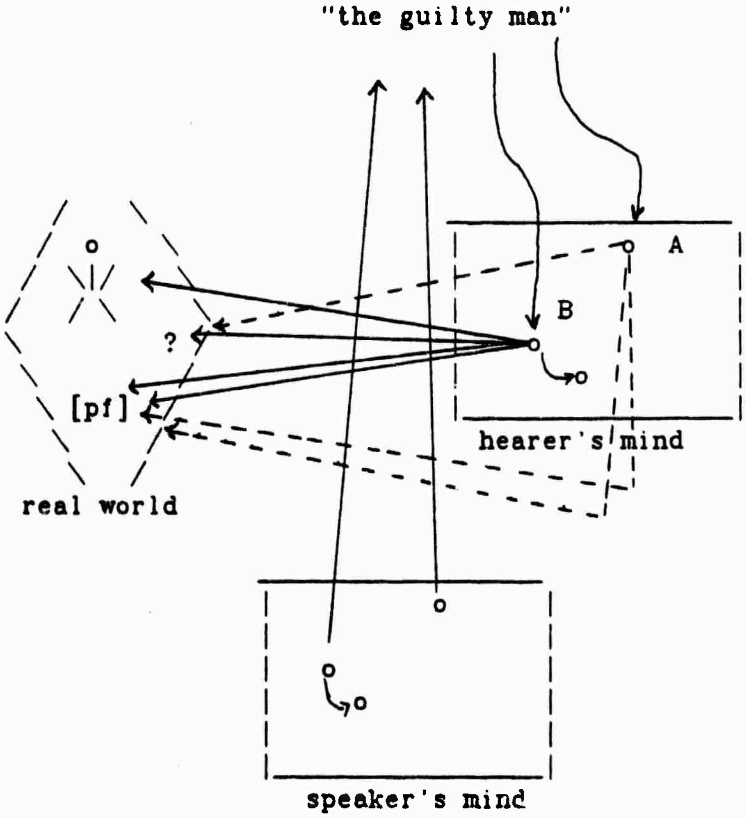
After the hearer represents an a-indefinite as a new mental item, s/he chooses from three possible mappings to the real world These correspond to the interpretations of "a woman professor I met" in D2-2a where the speaker has some one person in mind (though there may be others as well) but does not believe the hearer knows which one, the "at least one" or representative reading of "a new employee" in s2 and the generic reading of "an elephant" in s6. If the interpretation of "a guilty man" is that there is some one person the speaker had in mind, then the hearer must map the item A in figure 2 using the anonymous individual mapping. If "a guilty man" is used like "a new employee" above, then the mapping is a partial function that is to be evaluated at some later point, and if "a guilty man" is used as a generic, it is a different partial function that maps to a prototype that describes members of the generic class.

The speaker may have one additional mapping, namely to a particular individual. The hearer may actually have a previous item in memory which corresponds to the real world individual that the speaker has in mind when using an a-indefinite. For example, the speaker may say "Mary has a sneaky black cat." The hearer may already know all about Mary's black cat, but the speaker cannot expect the hearer to associate such information with the a-indefinite used simply because the speaker has not indicated his/her intention for the hearer to do so. Such intentions are expressed through uses of defnps.

Turning to the uses of defnps, consider figure 3 which is a variation on the previous two. Unlike a-indefinites, defnps may map into either a new item A in the hearer's mind or to an item B which specifies a previous item. Defnps are also used to create mappings of four different types, three of which are similar to those for a-indefinites.

The most common use of a defnp is the referential use. The defnp is meant to refer to some particular individual or item in the world. From the hearer's point of view, this use corresponds to mapping a mental item to a particular individual in the real world, as shown in figure 3. To assure such a map the hearer must have enough descriptions of the individual to distinguish it in any context. By having a canonical description, such as a name, this can be easily achieved. This type of knowledge will be called "referential knowledge of an individual." It is possible to have somewhat fewer descriptions and still map to an individual. However, if a speaker uses a defnp referentially with knowledge that the hearer has less than complete

FIG. 3. MAPPINGS FOR "THE GUILTY MAN"



information, the speaker runs the risk that the defnp will not map to the intended individual or to any individual at all.

A defnp that maps to a mental item may also be used to create the mapping for an anonymous individual. While this is like the mapping for the anonymous individual with a-indefinites, for defnps there is a commitment to the existence of only one such individual.

The anonymous individual napping can be seen as the other end of a continuum with the referential use. If the hearer has very little information about the item to which the defnp corresponds, s/he will be unable to choose a particular individual in the world to map the item into and may have to settle for a map that says there is such an individual even if it cannot be picked out. Midway between this circumstance and referential knowledge are the situations where a hearer has enough knowledge to map to an individual in some contexts but not in others. For example if X and Y are discussing "the woman over there," X has enough information to map the defnp onto some woman in the room, but X may not know that her name is Ms. Jones, that she is the sister of Ms. Smith. X will not be able to map the mental item corresponding to "the sister of Ms. Smith" to the woman in the room.

A mental item may also map as a partial function to be evaluated at a later time as in "The first president of the US had to be a Southerner"; the speaker is not saying that George Washington had to be what he was, but rather that, whoever it was that became president, had to have that trait. Another map to a partial function is the prototype map; this map is like the one for a-indefinites. It is also possible that no mapping at all is intended. For example, if a speaker says "The first president of the US could not have lied about his father's tree," it is possible to use the defnp without concern for whether it now maps to someone or ever will. The speaker's purpose may be to comment about the person, who as the first president, did or did not do something without regard for who that person could be.

When the speaker uses a defnp, his/her own real world mapping for the mental item to which the defnp corresponds may be different than the one that the hearer is intended to use. Reconsider s12 below.

s12 The first place winner of the Boston Marathon got a cash prize.

A speaker could have complete referential knowledge of the defnp when uttering s12, hence map the corresponding item into an individual, and yet expect only that the hearer treat the defnp as a description that maps onto some anonymous individual or as a partial function mapping.

A crucial question for defnp interpretation is whether the speaker's mapping matters to the hearer. If it did not, the hearer would have less information to consider in determining an interpretation (i.e. a mapping). As it turns out, the speaker's mapping does matter: without knowledge of it, the hearer will conclude that some utterances have peculiar intended meanings. Several cases need to be explored.

Suppose the speaker is about to construct the utterance "Who is the president of BBN?" and can use only an anonymous individual mapping for the item that corresponds to the defnp in his/her mind. The speaker believes however that the hearer can map to a particular individual. The hearer of the utterance maps "the president of BBN" to T. Jones. If the hearer believes the speaker is not aware of the particular individual, the question is a simple request for information about reference. Were the hearer to believe the speaker could map to the particular individual, the hearer would be forced to conclude that the speaker was asking a question for which s/he knew the answer. Such a question could make sense only if the speaker doubted the hearer had such knowledge and wanted to test the hearer.

A second circumstance where the speaker's mapping plays a role in the hearer's understanding can be exemplified with s13 repeated below.

s13 Go to the Alfred Building and see the woman in charge of of admissions.

The speaker may utter s13 having either an anonymous individual mapping, a partial function mapping or a particular individual mapping in mind. In the first instance, the speaker may have some additional information about the item s/he has chosen to describe in words as "the woman in charge of admissions" while not have referential knowledge. In the second instance the speaker may just believe there is some woman in that building in charge of admissions and who it may be will be found out at some later point (such as after going to the building). In the third instance the speaker has referential knowledge of the individual described by the noun phrase. At the same time the noun phrase may map to the hearer's mind as a new item and hence one for which the hearer has no referential knowledge. The hearer's problem is to decide which mapping the speaker intended him/her to use for the noun phrase.

The speaker could not have intended that the hearer use a particular individual mapping since the defnp maps as a new mental item (assuming the speaker is aware that the description is new for the hearer). The hearer may believe the speaker has referential knowledge for the noun phrase but as long as the hearer believes the speaker knows that the hearer does not have referential knowledge, the hearer can rule out being expected to make an individual mapping. The hearer is also aware that the speaker has some mapping (rather than none) because if the speaker had no mapping in mind, the command given in the utterance is vacuous. Were the hearer aware that the speaker had one or the other of a partial function map or an anonymous individual map, the hearer would have evidence for which way to interpret the noun phrase. Lacking that information the hearer can determine which was meant only by considering other aspects of the intended meaning of the utterance besides the noun phrase. Thus knowledge of the speaker's mapping can help discern the noun phrase interpretation.

In the example below, the speaker may use the noun phrase with a partial function mapping interpretation[1].

s14 The wife of the department head needs a tough skin and a pleasant smile.

Upon hearing the sentence, the hearer may recognize that the defnp maps to an item which specifies some previous mental item for which there is a map to a particular individual Mrs. Big. However, this is not the mapping the hearer is intended to use. The speaker, in saying s14, is stating something s/he believes true of whoever is the department head's wife, not just Mrs. Big The hearer has the burden of recognizing that s/he must not jump to conclusions about the individual map. When the hearer is aware that the speaker knows that there is an individual map to Mrs. Big, the hearer can use this knowledge to conclude that the individual map was not meant; were it intended, the speaker could have signaled it by use of "Mrs. Big" in place of the defnp and saved the defnp for the partial function mapping.

One final comment is needed regarding the distinction between anonymous individual and partial function mappings. A certain example, from Barwise and Perry, will illustrate the difference between them.

[1] The defnp use here corresponds the most closely to the value free/attributive uses discussed earlier.

Sherlock Holmes and Watson are trying to determine who murdered Jones. They know that someone did, and the evidence suggests that the person loved Mary. Thus Holmes can say "The murderer of Jones loves Mary." Barwise and Perry claim that the defnp is value free: it cannot be otherwise since neither Holmes nor Watson know the murderer, and so the defnp must be evaluated at some later time. In the expanded schema of mappings given in 3, there seems to be a different interpretation, namely "the murderer of Jones" will be interpreted with an anonymous individual mapping rather than a partial function mapping. Why? Both Holmes and Watson believe there is a murderer, and while they don't know who, they have more information in their memories than the belief that they will later find out who it is: they have evidence about the murderer. Hence the distinction in the two mappings comes simply from what other mental material is available besides that which is expressed in the noun phrase. Sometimes a hearer will not be able to distinguish which of the two mappings was intended because s/he will not have any knowledge of the speaker's mental material beyond that expressed in the noun phrase; yet when s/he does, it is relevant in determining just who was being spoken about.

I will not explore examples of all the pairings of speaker and hearer mappings as there are no notable characteristi beyond the ones already discussed. Now that I have described some of the situations of a speaker and hearer with respect to interpreting a noun phrase, I want to ask what kinds of knowledge the hearer uses to recognize the proper mapping. One kind has been explored, but there are several other relevant kinds. I shall also comment on how the recognition might take place, but this topic remains speculative until further research is undertaken.

9.4 Recognizing Uses of A-Indefinites

How might a hearer go about distinguishing the three uses of a-indefinites shown in figure 2? The hearer relies on a variety of factors to tease out the intended use, and some of these involve how the hearer's beliefs interact with the communicative purpose of an utterance. In the discussion that follows the role of beliefs and communicative purpose will be of fundamental concern. They distinguish the perspective taken here from earlier research in linguistics and philosophy.

To illustrate the factors involved, I will consider several examples that demonstrate that understanding is affected by the knowledge of the purpose of the utterance, of the speaker's overall goals and of the proper use of definite referring expressions as well as indefinites. I will also show how a hearer must use knowledge of everyday pragmatics and sentence semantics in understanding. Finally, some of these examples will illustrate cases where there is insufficient knowledge to determine which use the speaker intended.

s15 could be said in the situation where a user is talking to an assistant about a graphics display. On the screen is a map, and the travel patterns of ships, boats, tankers and barges are being presented so that the user can gather information about them.

s15 A ship icon should be blue.

The a-indefinite in s15 is usually taken to be a generic, and the sentence is interpreted as a standing order for how to present ships on the graphic display. Let us assume that the speaker's intention to issue a standing order is not fully recognized before the indefinite is understood. The hearer will know, simply from conventions of English usage, that the speaker intends the sentence to inform the hearer about the speaker's beliefs regarding contents of the sentence.[1] The hearer must determine what else the speaker intended.

To understand that the speaker intended to give a standing order, a hearer can hypothesize such an intention and then examine the indefinite to decide on its use, and its acceptability as part of the standing order. Now the speaker may have used the indefinite either because s/he had in mind mapping with an anonymous individual or a partial function. It is possible to rule out the individual reading for the standing order interpretation because if the speaker had an anonymous individual in mind, the nature of a standing order demands that s/he communicate exactly which individual (and hence use a defnp that expresses the referential nature of the description). Note that ruling out the anonymous individual mapping does not imply that the hearer has to be certain that a standing order is the only possible intention. Rather, finding an interpretation of the a-indefinite that is consistent with the hypothesized standing order as the speaker's intention lends support to that hypothesis. Other interpretations could be considered in parallel with understanding the utterance as standing order and the a-indefinite as a generic.

Once the anonymous individual mapping is ruled out, the he [illegible] ust still

[1] See Sidner & Israel [16] for a discussion of a theory and a model based on Grice's [8] work for recognizing such intentions.

determine which type of partial function (a representative or generic) mapping was intended. Making a representative icon blue would mean choosing some one from the screen (or elsewhere) and changing its color. While it is possible for a user to want such a thing, it is difficult to imagine how this would accomplish his/her overall task of gathering information about screen items. Since such an interpretation cannot be easily explained in terms of the user's goals, it is not likely to be the intended interpretation. The generic interpretation, however, presents no such difficulties; making the generic ship icon blue as a standing order is possible as a way of distinguishing screen objects in general. Hence the generic reading can be chosen as the interpretation of the a-indefinite in s15.

For this example, two factors have been crucial to the interpretation of the indefinite (in addition to proper syntactic and semantic interpretation, which is assumed here): a recognition of plausible speaker intentions, including a context of overall goals in which the intention can be explained, and an understanding of the effects of both definite and indefinite descriptions to distinguish referential use. The first of these is a new finding about speaker intentions and a-indefinites. That speaker intentions are crucial to referential interpretations of defnps is no surprise given the work of Perrault and Allen [13]. That the intentions must fit within a framework of overall goals has also been explored by Perrault and Allen and Sidner and Israel. However, the relation of intentions and indefinites suggests that indefinites must also be handled in theories of interpreting speaker intentions.

Understanding the effects of using definite descriptions can be explained by two assumptions to be used when reasoning about noun phrases:

1. A speaker who uses a particular individual mapping when creating a description and who expects the hearer to use one as well for that description must use a definite reference for the description.
2. A speaker can only use a definite description and a particular individual mapping, when s/he is certain that the description as argument to the mapping cannot be used to pick out several objects.

Assumption 2 is a corollary to Grice's [9] maxim of being perspicuous and avoiding obscurity. From the hearer's point of view, it insures that if a defnp is used, and the hearer believes several objects could be so described, then either there is an error in the communication channel, or the speaker intended the defnp to be used some other way (such as generically—with an implied universal quantifier, or with deixis).

In D6, U is a user of a message system and is looking at a list of message names that includes a summary line of the message contents and its author. S is the controller of the message system and knows how to carry out U's requests.

D6-1 U: Send message17 to Jones.
2 S: You can't send <u>a</u> <u>deleted</u> <u>message</u>.
3 S: You can undelete it and then send it.

The use of the a-indefinite in D6-2 is more complicated than the one in s15 because it depends in part on alternate semantic interpretations, in particular, on the scope of negation. The scope could be interpreted in any of the ways below:

```
~ (Ex:a deleted message (You can send x))
  Ex:a deleted message ~ (You can send x)
  Ex:a deleted message (You ~(can) send x)
  Ex: a deleted message (You can (~send) x)
```

Whichever scope is meant, the pragmatic interpretation of the indefinite might be to any of the three mappings in figure 2,[1] and two of them must be ruled out.

Let us begin unwinding the interpretation by asking whether the speaker could have intended an anonymous individual mapping for D6-2. The anonymous individual the speaker has in mind cannot turn out later to be message17. Why? Message17 is mutually known to both the speaker and hearer, but a-indefinites are for use when the object the speaker has in mind is not known to the hearer. If message17 were what the speaker wanted to talk about, it is misleading to use an a-indefinite. This explanation hinges on yet another Gricean based assumption about noun phrases: Avoid misleading the hearer by using an a-indefinite for objects both speaker and hearer know of.

Suppose, however, that the speaker followed this rule and instead had some other deleted message in mind, one that neither speaker nor hearer can fully describe. For that a-indefinite description, the anonymous individual mapping for the a-indefinite would be appropriate However, the speaker's intentions for both D6-2 and D6-3 are unintelligible with that interpretation.

[1] For a discussion of scope of indefinite noun phrases, especially inside of other quantifiers, see Webber [21]. Such surface a-indefinites seem to be governed by rules of quantification and hence fall outside the discussion being pursued here.

Turning to the partial function interpretations, A must ask whether B is saying that A can't send some (i.e. representative) deleted message or a prototypic generic one. Since A can't send prototypic messages in any case, such an interpretation is unlikely. Which leaves only the representative reading, no matter what semantic scopes were given. This interpretation is reasonable, since, whatever the scope, if a representative message can't be sent, then message17 cannot be either. Hence B's intention to communicate the limits on deleted messages in general reflects a goal to communicate how it will respond to requests regarding message17.

The preceding explanation of the interpretation of D6-2 parallels much of the explanation of s15 because it draws upon the hearer's knowledge of how referential and indefinite noun phrases are used, and what the speaker's intentions are. In both explanations I have described processes for how the a-indefinite gets understood, but I do not want to make claims about the specifics of such processes. What I have suggested is that the hearer considers readings and rules them out. This is only one explanation, and there is insufficient evidence at this time to exclude others. However the hearer actually comes to the intended interpretation, we must first recognize the types of knowledge that are needed for the hearer to make his/her decision.

To close the discussion of a-indefinites I want to return to the examples s16 and D7-2b given at the beginning of this paper.

s16 A new employee will be serving on the labor relations committee.

D7-1 While I was at MIT, I went to the AI lab.

2 b)I'd like you to meet a woman professor I met there.

Suppose s16 was said by one friend to another in a conversation. Only the generic reading is nonsensical (one cannot have generic employees about to serve on committees), but nothing in the speaker's intention makes clear whether the speaker is talking about a representative employee or a particular one about whom there is no referential knowledge (i.e. an anonymous individual mapping). Nor without additional information about the nature of the discussion and the two discussants' histories, is it possible to recognize a set of goals for the speaker that would indicate why s/he might talk about the anonymous rather than representative employee. Hence such an example is ambiguous between two readings for the a-indefinite. In contrast, in D7-2b, the a-indefinite in isolation could be used with a representative partial function mapping, but in the context of the speaker wanting the hearer to meet a person so described, the phrase must be taken as mapping to an anonymous individual.

While the two sets of examples, s16 and D7, and s15 and D6, can be contrasted for the way the speaker's intention eliminates certain readings of the indefinites, both sets of examples depend on hypotheses about the speaker's intention, on knowledge of how indefinites are used, and on knowledge of sentence semantics and of the pragmatics of the everyday world. When a relation between the speaker's intention and some more general goals can be deduced, this information is crucial to eliminating certain readings. Generally without knowing both the speaker's intention and his/her more general goals, ambiguous readings of a-indefinites result. All the examples share in common a dependence on knowledge of sentence semantics and pragmatics of the everyday world.

To summarize then, I have shown that the speaker's intention conveyed in an utterance containing an a-indefinite, and the overall goals of the speaker are significant to interpreting the a-indefinite. In addition, assumptions about how referring expressions are used, including the role of Grice's conversational maxims, knowledge of everyday pragmatics and sentence semantic information all contribute to distinguishing readings of an a-indefinite. As we shall see, these same factors come to bear in interpreting defnps.

9.5 Interpreting Defnps

When a hearer comes to interpret a defnp, s/he must decide how it was used; given figure 3, we can see that the hearer must differentiate among many possible pairs of mappings (pairs of the speaker's and hearer's mappings to the real world), if a differentiation is possible at all. Each use described in the figure has a distinct intended effect on the hearer.

The speaker is constrained to use a defnp with only certain expectations about how the hearer can make a mapping. Suppose the speaker knows that the hearer is mapping the defnp to a new item in memory. Then the speaker, depending on his/her own mapping to the real world, can intend only certain mappings to the real world to be discernible by the hearer. First, if the speaker maps with a particular individual mapping, s/he can expect the hearer to use either an anonymous individual mapping or a partial function mapping. A single exception to this circumstance is the presence of deixis. If the speaker says "May thinks <u>the lady in black</u> came with John," while looking in the direction of a woman so dressed, the speaker is using more than the defnp to make a mapping available to the hearer, and hence can intend a particular individual mapping even when the defnp corresponds something new in memory. Some uses may seem to be particular individual mappings as in "Pick up the big red block," which can be said to a hearer who has no previous knowledge of the block. However, the use is really a partial function mapping that is intended to be evaluated in some appropriate immediate context (such as the table in front of the hearer) rather than at some future point in time.

For a new item in the hearer's memory, the speaker may have an anonymous individual mapping and can expect only an anonymous individual or partial function mapping of the hearer. When the speaker has a partial function mapping, s/he can only expect such a mapping of the hearer. The same is true of generic mappings. To expect any others would mean that the speaker believes the hearer has some additional information available, but since the defnp corresponds to a new item, the speaker cannot expect the hearer to have additional information.

A similar analysis for the speaker's expectations results from considering mappings available to the hearer when the defnp maps to an item that specifies a previous memory item. Depending on whether the previous item already includes a real world mapping of one type or another, the speaker must behave in different ways.

Suppose the previous memory item does include a mapping (and the speaker knows which). The speaker can expect the hearer to interpret the defnp using whichever non-generic mappings the hearer has when the speaker's own mapping is either an anonymous or a particular individual mapping (just which depends on other knowledge availabie to the hearer), because the hearer has sufficient information to make the mapping. When the speaker's own mapping is one of the partial function mappings, the speaker can only expect the hearer to use one of these. Were the hearer to use another mapping, the two people would be talking about different things. For example, if the speaker uses "the murderer of Jones" with a partial function map (and hence does not know who the murder is or much about him), while for the hearer "the murderer of Jones" maps to a person John Smith, the two people will reach different conclusions about whatever is said of the murderer

When the previous item in the hearer's memory does not include a real world mapping, the speaker's expectations can differ. The speaker can expect the hearer to use only those mappings with the same or less information than the one the speaker uses. For example, the speaker cannot have an anonymous individual mapping for a defnp and expect the hearer to map to a particular individual. The hearer may be able to do so, but the speaker can be assured of this possibility only by knowing the contents of the previous memory item and hence actually having enough information for a particular individual mapping.[1]

While the speaker may be constrained in his/her mapping choices the hearer must discern which was intended. When the hearer has perfect knowledge of the speaker's own mapp gs as well as what the speaker thinks the hearer's

[1] This situation is not equivalent to the "who is the President of BBN" case because in that circumstance the speaker believes the hearer has a previous memory item that includes a mapping to a particular individual.

mappings are, the hearer's job is trivial; s/he can follow the outline above. But usually hearers have much less information. They may know that the speaker has a certain kind of mapping but not know what the mapping produces or they may not know the kind of mapping at all.

In general the hearer uses whatever knowledge s/he has about the speaker as well as about pre-existing specifications for a defnp. In addition the hearer must consider the speaker's meaning and intentions for the whole utterance and his/her own pragmatic knowledge that is relevant for that utterance.

To make this discussion more concrete, let us consider a number of examples in which I will tease out the distinctions between what the speaker and hearer believe, and describe how the hearer's beliefs about speaker's intentions affect the interpretation of the defnp.

In D8 the hearer may first decide that the speaker has referential knowledge of "the handsome man" (otherwise the speaker could not have met or seen him), but not know who the speaker takes the defnp to refer to.

D8-1 I saw <u>the handsome man I met at Clarence's</u> again today.
2 Since you know him, could you call him and ask him to dinner?

When the hearer turns to consideration of his/her own beliefs about the defnp, two situations can occur. First suppose that upon searching memory the hearer discovers a specification that describes the handsome man and gives referential information about who he actually is. Therefore the hearer can assume the defnp is to be mapped as a particular individual. This interpretation is consistent with the rest of the discourse where the speaker asks the hearer to call him up, something that requires referential abilities.

An alternative situation occurs when upon searching memory, the hearer discovers only a specification that notes that in the past the speaker met someone handsome at a soiree at Clarence's and told the hearer about him; the hearer never learned who this person was or met him. The hearer can conclude that the defnp is meant to be mapped with an anonymous individual mapping since the speaker should have reliable beliefs about what the hearer learned during their discussion. When D8-2 is encountered, the hearer can easily recognize the proper co-specification of "him,"[1] but will then be in a quandary since the speaker has said that the hearer knows this handsome man. For that to be true, the hearer must have referential knowledge associated with the specification. Hence, the hearer can conclude that the speaker intended the hearer to use a particular individual mapping for the defnp in D8-1. Since it is not true that the hearer has such referential knowledge of the handsome man, the hearer may decide that s/he must tell the speaker about their miscommunication or pick some other way to set the speaker straight about the hearer's knowledge of the handsome man.

Suppose D8-2 had been "Could you call him and invite him to dinner?" with no statement about knowing such a person. The defnp in D8-1 might be interpretable as any of the non-generic mappings. For the partial function of anonymous individual mappings, the speaker must intend for the hearer to use the description to locate the person described at some time in the future. An open question is whether a pragmatic theory ought to stipulate conditions of adequacy of a description for locating the object referred to; see Goodman [7]

[1] Accounts of this process are given by both Sidner [18] and Reichman [15].

for issues on failures to adequately describe an object. This matter will not be settled here.

A contrasting use of the defnp is given in D9-2a.

D9-1 While I was at MIT, I went to the AI lab.
2 a) I'd like you to meet the woman professor I met there.

While the hearer's reasoning about the defnp interpretation for the speaker and hearer is similar, the speaker's intention is different. The speaker is informing the hearer of his/her desire to have the hearer meet a particular person, and may also be doing so as part of beginning a social introduction. Given assumption 2 discussed before, the defnp use also conveys the fact that there is only one woman professor that fits the speaker's description. Hence the defnp use is either an anonymous individual or partial function mapping, depending upon the hearer's actual knowledge.

It is possible for a defnp use to be ambiguous among all the mappings. In the sample below, after A's first statements, it is possible that A has used the defnp in correspondence with any one of those mappings and B may not be able to decide which.

D10-1 Speaker A: The second musician is scheduled to go on at 11.
2 He will play for 15 minutes.
3 Speaker B: Oh, no, He'll play for longer than that.
4 He's a real egotist.
5 Speaker A: Oh, do you know him?

B may decide (perhaps incorrectly) that A knows B has referential knowledge of musician #2, and reduce the choice to a particular individual mapping. Such a decision would be in keeping with B's comments in D10-3 and 4. Only at D10-5 would B discover his error.

Ambiguities such as these need not be resolved totally for effective communication. In fact by the end of the exchange it is unclear whether A has referential knowledge of the musician. Bobrow and Webber [2] and van Lehn [19] have proposed retaining ambiguity until it must be eliminated when interpreting quantifier scope, and examples in this paper suggest that the interpretation of noun phrases may be understood in a similar way. Naturally such a model will change the manner in which computational natural language systems go about finding references.

Generic defnps present the greatest difficulty for theories of defnps because it is difficult to state what knowledge the hearer brings to bear to distinguish such uses. A representative set of perplexing examples of generics is given below, in addition to s7:[1]

s17 The Incas did not know of the wheel.

s18 * Monkeys do not use the instrument.

s19 The Rhodesian government prevents the dissident from leading a normal life.

s20 The Rhodesian government caught the dissident. <not a generic defnp>

s21 The teenager in the average US school reads below his potential.

s22 That is the blue nosed mongoose.

s23 The Chinese regime forces the country's newspaper to serve only as parrot of its views. <not a generic defnp>

In all the examples above the hearer must determine whether the speaker is intending to be informative about a class of objects (and thus speaking

[1] The first two of these are from Vendler [20].

generically) or about only one. While it is not easy to specify exactly what such an intention comes to, I can clarify some of the knewledge needed to discern such in*entions.

For examples such as s7 and s21, the hearer must recognize that the speaker is being informative and is characterizing an object described by the defnp· the hearer must also discern that the defnp by itself can describe any of several objects. With those two facts and assumption 2, the hearer can recognize that the speaker is speaking generically with the defnp about a prototypical member. Some sentences set a generic tone with cther kinds of noun phrases (such as s18); defnps within a generic sentence can be interpreted generically, even when the resulting interpretation is ungrammatical.

As one would expect there are constraints on whether a defnp can be interpreted as a generic, either in a generic sentence or in one that is not apparently generic. The hearer must have knowledge that there is a class of objects fitting the description in the defnp. Thus for "the wheel," "the dissident," and "the blue nosed mongoose," the hearer may easily discern a class. But for "the instrument" and "the country's newspaper," no such class is evident. In a generic sentence, a defnp that does not describe a prototype of a natural class gives an odd reading (as with s18), but in a sentence that can be interpreted non-generically (s23), the defnp serves as a specific.

Another constraint on the generic defnp is that the sentence semantics must permit a generic reading to go through. In s20, the generic reading of "the dissident" expresses a possible class, but it is not possible for a government to have caught a prototype in that class. Just how to characterize sentence semantics in general is a non-trivial matter since metaphorical uses of certain verbs will permit generic readings of their objects while others will not.

The previous collections of examples, while not exhaustively illustrating all the cases of defnps, describe some strategies a hearer might use in interpreting defnps and demonstrate what knowledge s/he must factor into those strategies. I have also shown that defnp interpretation relies on the same kinds of information as a-indefinites, namely,

- the speaker's intentions,
- the speaker's overall goals,
- assumptions about use of referring expressions as a reflection of Gricean maxims,
- beliefs of the speaker and hearer, including knowledge of everyday pragmatics,
- sentence semantic information.

9.6 Looking Ahead

In this paper I have explored the kinds of pragmatic knowledge that a hearer needs to use to understand the speaker's intended interpretation of a noun phrase. I have presented a computational viewpoint on these matters by exploring the interactions between kinds of knowledge and the manner in which the hearer takes into account the speaker's role in the communication.

This paper is limited in its scope since it only provides a framework for describing what knowledge a hearer brings to bear. It does not detail the form or content of such knowledge, suggest how to reason about beliefs (which is complicated by a number of unresolved problems in epistemology) nor crucially does it account for *how* hearers use their knowledge. I assume that research on these topics lie ahead, and that it will tell us more about the pragmatics of non-anaphoric noun phrases.

REFERENCES

[1] Barwise, J. and Perry, J. The Situation Underground. Unpublished manuscript, Stanford University, August, 1980

[2] Bobrow, R.J. and Webber, B.L. PSI-KLONE - Parsing and Semantic Interpretation in the BBN Natural Language Understanding System. CSCSI/CSEIO Annual Conference, CSCSI/CSEIO, 1980.

[3] Cohen, P.R. The need for identification as a planned action. Proceedings of the International Joint Conference in Artificial Intelligence, The International Joint Conferences on Artifical Intelligence, Vancouver, B.C., August, 1981, pp. 31-36.

[4] Dahl, O.. *On Generics*. Cambridge University Press, New York, 1975.

[5] Donnellan, K.S. "Reference and definite description." *The Philosophical Review 75* (1960), 281-304. Reprinted in Steinberg & Jacobovits (Eds.), Semantics, Cambridge University Press, 1966.

[6] Donnellan, K. S. Speaker Reference, Descriptions and Anaphora. In Cole, P., Ed., *Syntax and Semantics: Volume 9, Pragmatics*, Academic Press, New York, 1978, pp. 47-68.

[7] Goodman, B.A. *Generating Plans from Natural Language Instructions*. Ph.D. Th., University of Illinois, forthcoming.

[8] Grice, H.P. "Utterer's Meaning and Intentions." *Philosophical Review 68* , 2 (1969), 147-177.

[9] Grice, H.P.. *Logic and Conversation*. Academic Press, New York, 1975.

[10] Grosz, B. J. The representation and use of focus in dialogue understanding. Technical Report 151, Artificial Intelligence Center, SRI International, Menlo Park, California, July, 1977.

[11] Grosz, B. J. Utterance and objective: Issues in natural language communication. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, IJCAI, 1979, pp. 1067-1076. Reprinted in AI Magazine, American Assocation for Artificial Intelligence, 1, 1, Spring, 1980

[12] Lyons, J.. *Semantics Volume 1*. Cambridge University Press, New York, 1977.

[13] Perrault, C.R., and Allen, J.F. "A plan-based analysis of indirect speech acts." *American Journal of Computational Linguistics 6*, 3 (1980), 167-182.

[14] Perrault, C.R. and Cohen, P.R It's for your own good: A note on inaccurate reference. In *Elements of Discourse Understanding*, Joshi, A., Sag, I., & Webber, B., Eds., Cam`ridge University Press, Cambridge, Mass., 1981.

[15] Reichman, Rachel. Plain Speaking: A Theory and Grammar of Spontaneous Discourse. Tech. Rept. BBN Report 4681, Bolt Beranek and Newman Inc., 1981.

[16] Sidner, C.L., and Israel, D.J. Recognizing intended meaning and speaker's plans. Proceedings of the International Joint Conference in Artificial Intelligence, The International Joint Conferences on Artifical Intelligence, Vancouver, B.C., August, 1981, pp. 203-208.

[17] Sidner, C.L. Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse. Tech. Rept. Technical Report 537, MIT Artificial Intelligence Laboratory, 1979.

[18] Sidner, Candace L. "Focusing for the Interpretation of Pronouns." *American Journal of Computational Linguistics 7*, 4 (October-December 1981), 217-231.

[19] Van Lehn, K. Determining the Scope of English Quantifiers. Tech. Rept. Technical Report 483, MIT Artificial Intelligence Laboratory, 1978.

[20] Vendler, Zeno. *Linguistics in Philosophy*. Cornell University Press, Ithaca, N.Y., 1967.

[21] Webber, B.L. A formal approach to discourse anaphora. BBN Report 3761, Bolt Be"anek and Newman, Cambridge, Massachusetts, May, 1978.

10. REPAIRING MISCOMMUNICATION: RELAXATION IN REFERENCE[1]

Bradley A. Goodman

10.1 INTRODUCTION

In natural language interactions, a speaker and listener cannot be assured of having the same beliefs, contexts, backgrounds or goals. This leads to difficulties and mistakes when a listener tries to interpret a speaker's utterance. One principal source of trouble is the description constructed by the speaker to refer to an actual object in the world. The description can be imprecise, confused, ambiguous or overly specific; it might be interpreted under the wrong context. This paper explores the problem of resolving such reference failures in the context of the task of assembling a toy water pump. We are using actual protocols to drive the design of a program that plays the part of an apprentice who must interpret the instructions of an expert and carry them out. A primary means for the apprentice to repair such descriptions is by relaxing parts of the description.

Consider the dialogue below which exemplifies some kinds of complex descriptions used in utterances. Here A is instructing B to assemble part of

[1] The following is a revised version of a paper [13] given at the 1983 National Conference on Artificial Intelligence (AAAI 83). Other related papers include [11, 12].

a toy water pump [14, 9, 10]. Refer to Figure 1 for a picture of the pump. A and B are communicating verbally but neither can see the other. (The bracketed text in the excerpt shows what was actually occurring while each utterance was spoken.) Notice the complexity of the speaker's descriptions and the resultant processing required by the listener. In Line 1, B interprets "the long blue tube" to refer to the STAND. When A adds thc relative clause "that has two outlets on the side," B is forced to drop the STAND as the referent, to relax the color "blue" to "violet," and to select the MAINTUBE. In Line 6, A's description "the nozzle-looking piece" fits more than one object and B selects the NOZZLE instead of the SPOUT. The speaker was sloppy here because he didn't know the word "spout." A's addition of "the clear plastic one" in Line 7 rules out the NOZZLE - which is red and opaque - in favor of the SPOUT. Line 16 demonstrates a case where A previously focused B's attention on one object and intends to switch that focus to another one. In this case, B doesn't shift focus. This lack of agreement on what is in focus leads to confusion later on in the dialogue.

```
A:  1.  Take the long blue tube
                    [B reaches toward STAND]
    2.  that has two outlets on the side -
                    [B takes MAINTUBE]
    3.  that's the main tube.
    4.  Place the small blue cap
                    [B takes CAP]
    5.  over the hole on the side of that tube.
                    [B pushes CAP on OUTLET1]
    6.  Take the nozzle-looking piece,
                    [B grabs NOZZLE]
    7.  the clear plastic one,
                    [B takes SPOUT]
    8.  and place it on the other hole
                [B identifies OUTLET2 of MAINTUBE]
    9.  that's left, so that the nozzle
   10.  points away.
```

[B installs SPOUT on OUTLET2 of MAINTUBE]
11. Okay?

B: 12. Okay.

A: 13. Now take the blue lid type thing
[B takes TUBEBASE]
14. and screw it onto the bottom
[B screws TUBEBASE on MAINTUBE]
15. ooops,
[A realizes he has forgotten to have B put SLIDEVALVE into OUTLET2 of MAINTUBE]
16. undo the plastic thing
[B removes TUBEBASE but A meant the SPOUT]

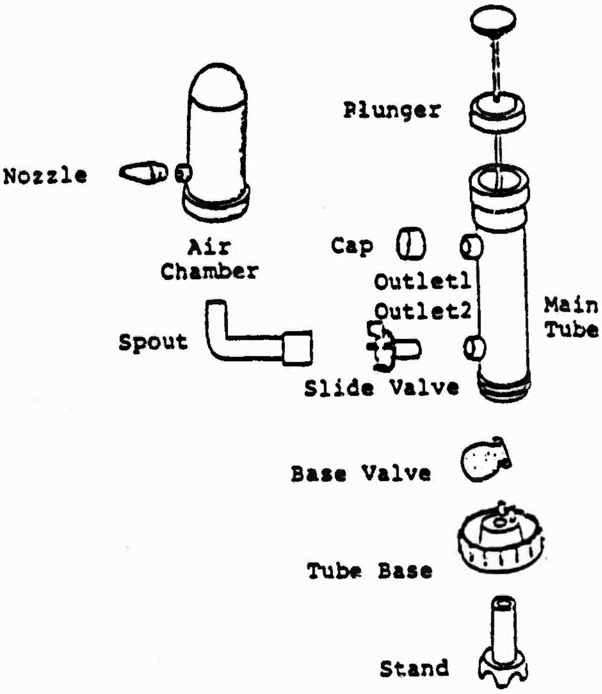


FIG. 1. THE TOY WATER PUMP

In conversation people use imperfect descriptions to communicate about objects; sometimes their partners succeed in understanding and occasionally they fail. I am working on a taeory of the use of extensional descriptions that will explain how people successfully use such imperfect descriptions.

One means of making sense of an approximate description is to relax portions of it that don't match objects in the hearer's world. Relaxation then is a form of communication repair [6] that hearers can use. As part of my work I am developing a reference identification module for a natural language system that will treat descriptions as approximate. It can relax a description in order to find a referent when the literal content of the description fails to provide the needed information. In this paper I will describe the relaxation component of the reference identification module and illustrate some of the sources of knowledge that guide it in relaxing a description.

10.2 THE DOMAIN

We are following the task-oriented paradigm of Grosz [14] since it 's easy to study (through videotapes), places the world in front of you (a primarily extensional world), and limits the discussion while still providing a rich environment for complex descriptions. The task chosen as the target for the system is the assembly of a toy water pump [9, 27, 10].[1] The water pump is reasonably complex, containing four subassemblies that are built from plastic tubes, nozzles, valves, plungers and caps that can be screwed or pushed together. A large corpus of dialogues concerning this task was collected (see [9]). These dialogues contained instructions from an "expert"

[1] This domain was chosen over the KL-ONE-Ed graphics editor domain because a large corpus of protocols were previously collected [9, 27]. We believe that the reference identification mechanism developed is transferable to the KL-ONE-Ed domain because both domains refer to real objects that are rich in perceptual features and have similar physical actions that can be performed on them.

to an "apprentice" explaining the assembly of the pump. This domain is rich in perceptual information, allowing for complex descriptions of elements in it. The data provide examples of imprecision, confusion, and ambiguity as well as attempts to correct these problems.

10.3 THE KNOWLEDGE TRANSFERRED IN AN UTTERANCE

In a task-oriented domain there is limited shared knowledge between speaker and listener (i.e., less than other domains since usually one participant knows a lot more about the task than the other). The underlying context is the achievement of the task. This requires a transfer of knowledge from the speaker - who is explaining how to perform the task - to the listener - who is to perform the task. The listener, thus, is building up knowledge (which becomes shared knowledge, i.e., mutually believed [7, 21, 16, 20]) from the speaker's utterances while attempting to perform the task.

At least two kinds of knowledge are conveyed in an utterance. For this paper I will focus on task knowledge [14] and communicative knowledge [25, 8, 22, 2, 3, 18]. Task knowledge is knowledge about the specific domain. It has three aspects: (1) the objects, the set of parts available to accomplish the task (the Real World); (2) the actions, the set of physical actions available to the listener; and (3) instructions linking objects and actions together to achieve some goal. Communicative knowledge consists of speech acts, communicative goals, and communicative actions. Speech acts are underlying surface forms that are performed by the speaker in making an utterance (e.g., REQUEST, INFORM) [25, 8, 2]. Communicative goals reflect the structure of the discourse (e.g., setting up a topic, clarifying, or adding more information [3]) and express how the utterance is to be understood and hence how the task it examines is used. A communicative act is a way of expressing the communicative goal that one wants to convey (e.g., communicate the goal, communicate the object's description, communicate the action). Only some of the possible communicative acts may be reasonable at any one time to accomplish the current communicative goal [23, 3, 18].

Things can go awry during communication. Trouble can occur due to either the way the information was transferred or the content of what was transferred. Problems can occur with the task knowledge: (1) the listener has a different view of the task than that of the speaker, (2) the listener is considering a different subset of objects than the speaker, or (3) the listener is considering a different subset of actions than the speaker. Difficulties with communicative knowledge are also possible. The speaker may use the wrong speech act (e.g., utters something (inadvertently) that would be conventionally interpreted as an INFORM when meant as a REQUEST) or the listener errs when interpreting the speaker's intention. In both cases it is the effect of the speech act that causes the trouble since it influences what the listener will do with what was said (i.e., what are the proper responses). Finally, communicative knowledge can cause mistakes and confusion if the listener and speaker differ on the communicative goal. They will feel they are communicating at cross purposes - leading to frustration.

10.4 THE KINDS OF PROBLEMS

Part of my research has been an examination of how a listener discovers that a repair of a description is needed, and how the listener di-covers the source of the problem in the communication.

- How the problems are discovered:

  1. The listener finds no Real World object to correspond to the speaker's description;
  2. the listener finds other than the requested number of Real World objects (i.e., too many or too few);
  3. the listener cannot perform the action specified by the speaker because of some obstacle; or
  4. the listener performs the action but does not arrive at its intended effect.

- Where the problems may reside:

  1. In the speaker's description of an object presented in the utterance;
  2. in the speaker's description of a physical action presented in the utterance;
  3. with the set of Real World objects that have been brought into attention (the speaker's set may differ from the listener's set);
  4. with the set of Real World actions that have been brought into attrntion (the speaker's set may differ from the listener's set);
  5. in the interpretation of the underlying force of the utterance (e.g., does the speaker want the listener to simply note the information in the utterance or to use it to do something); or
  6. with the hearer's concentration (e.g., the hearer may fail to

pay attention, missing or mishearing a word or the like).[1]

These observations signal conditions in which a mistake might occur and where it might be found. We will now explore what a listener has available for resolving miscommunication.

10.5 KNOWLEDGE FOR REPAIRING DESCRIPTIONS

When things go wrong during a conversation, people have lots of knowledge that they bring to bear to get around the problem (see [24]). Much of the time the repairs are so natural that we aren't conscious that they have taken place. At other times, we must make an effort to correct what we have heard, or determine that we need clarification from the speaker. This repair process involves the use of knowledge about conversation, its social conventions and the world around us.

In this work, I chose to consider the repair of descriptions rather than complete utterances. The most relevant knowledge for repair depends on the conversation itself and the Real World described therein. There are numerous sources of knowledge to consider that drive the reference repair process. Linguistic knowledge is the use of the structure and meaning of a description. Perceptual knowledge is a person's abilities to distinguish feature values, one's preferences in features by considering which seem more important (with respect to the person and the domain), and one's perception of an object.

[1] I am including this kind of problem because I have been talking about human dialogues. I will not, however, pursue it any further.

Discourse knowledge has to do with notions of the flow of conversation and its effects on highlighting relevant parts of the world. Hierarchical knowledge is concerned with generality and specificity information to determine if a description is too vague or too specific. Trial and error knowledge is information about how well a requested action succeeds (as specified) on the requested objects. Other knowledge sources, such as pragmatic knowledge [8, 2, 21, 4], and domain knowledge [14] will not be covered here.

10.5.1 Linguistic Knowledge in Reference

Different linguistic structures can be utilized to describe objects in the extensional world. This section outlines some of these structures and their meanings and shows how they can be used to guide repairs in the description.

A description of an object in the extensional world usually includes enough information about physical features of the object so that listeners can use their perceptual abilities to identify the object. Those physical features are normally specified as modifiers of nouns and pronouns. The typical modifiers are adjectives, relative clauses (adjective clauses) and prepositional phrases (adjective phrases). They are often interchangeable; that is, one could specify a feature using any of the modifiers. One modifier, however, may be better suited for expressing a feature than another.

Relative clauses are well suited for expressing complicated information since they are separate from the main part of the noun phrase and can be arbitrarily complex themselves.

- Complex relationships such as spatial relations (e.g., "the blue cap *that is on the main tube*"), and function information (e.g., "the thing with the wire *that acts like a plunger*").

- Assertions of "extra" information, information possibly outside the domain knowledge and not useful for finding the referent at this time. (e.g., "an L-shaped tube of clear plastic *that is defined as a spout*").

- Material useful for confirming that the proper referent was found. (e.g., "the long blue tube *that has two outlets on the side*").

- A respecification of the initial description in more detail. For example, in the case of the descriptions "the thing *that is flared at the top*" and "the main tube *which is the biggest tube*," the relative clauses are needed because the initial descriptions are too vague.

Prepositional phrases are better fitted for simpler pieces of information. They are often part of expressions of predicative relationships.

- A comparative or superlative relation (e.g., "the smallest of the red pieces"),

- A subpart specification - used to access the subpart of the object under consideration (e.g., "the top end *of the little elbow joint*," "that water chamber *with the blue bottom and the globe top*").

- Most perceptual features (e.g., "with a clear tint," "with a red color").

Just like relative clauses, prepositional phrases can also provide confirmation information.

Adjectives are used to express almost any perceptual feature - though complex relations can be awkward. Usually they modify the noun phrase directly, but sometimes they are expressed as a predicate complement. In those situations, the complement describes the subject of the linking verb (e.g., "the tube is *large*"). As with some of the relative clauses above, predicate complements have an assertional nature to them.

10.5.2 Relaxing a Description Using Linguistic Knowledge

The relaxation process attempts to relax features in a description in the order: adjectives, then prepositional phrases and finally relative clauses and predicate complements. This order was chosen by examining the water pump protocols and by noting where the linguistic forms come into play during reference resolution. Adjectives and prepositional phrases play a more central role while relative clauses usually play a secondary role during referent identification. Relative clauses and predicate complements exhibit an assertional nature that reduces their usefulness for resolving the current reference (whereas the information they express can be useful in subsequent references). The head noun can also be relaxed. It normally is relaxed last but could be relaxed prior to a relative clause (especially in the instances where the relative clause expresses confirmational information).

For example, consider the description "the large violet cylinder that has two outlets." Here, the features size, color and shape are described in the adjectives and head noun of the description, and the two subparts' function in the relative clause. Following the above rules, the relaxation of size, color and shape should be attempted before either the number of subparts or the subparts' functions. The relaxation order is influenced by the other knowledge sources so the order proposed here is not hard and fast.

10.5.3 Perceptual Knowledge in Reference

A major factor involved here is how people perceive objects in the world and how this can be simulated in my system. Each object for my system is denoted by two forms. a spatial (3-D) representation and a cognitive/linguistic form that shows how the system could actually talk about the object. The spatial description is a physical description of the object in terms of its dimensions, the basic 3-D shapes composing it, and its physical features [1]. The cognitive/linguistic form is a representation of the parts and features of the object in linguistic terms. It overlaps the spatial form in many respects but it is more suggestive of the listener's perceptions. The cognitive/linguistic form describes aspects of an object such as its subparts by its position on the object ("top", "bottom") and its functionality ("outlets", "places for attachment"). More than one cognitive/linguistic form can refer to the same physical description. Some properties of an object differ in how they are expressed in the two forms. In the 3-D form, there are primarily properties such as numerical dimensions (e.g., "3 feet by 5 feet") and basic shapes (e.g., generalized cylinders), while, in the cognitive/linguistic form, there are relative dimensions (e.g., "large") and analogical shapes (e.g., "the L-shaped tube").

Perception, hence, may involve interpretation. This can lead to discrepancies between individuals. People usually agree on the spatial representation but not necessarily on the cognitive/linguistic description and this can lead to problems. For example, misjudgements by the speaker in calling an object "large" can cause the hearer to fail to find an object in the visual world that has dimensions that are perceptually "large" to the listener.

To avoid confusing the listener, a speaker must distinguish the objects in the environment from each other. The perceptual features of an object provide people with a way to discriminate one object from another. A speaker must take care when selecting from these features since they can induce their own confusion. Perceptual features may be inherently confusing because a feature's values are difficult to differentiate (e.g., is the tube a cylinder or a slightly tapering cone?). They may also be confusing because the speaker and listener may have differing sets of values for a feature (e.g., what may be blue for someone may be turquoise for another). These characteristics affect the salience of a feature (see [19]) which in turn determines the feature's usefulness in a description. A feature that is common in everyday usage (e.g., color, shape or size) is salient because the listener can readily distinguish the feature's possible values from one another. Of course, very unusual values of a feature can stand out, making it even easier to discriminate a unique object from all other objects [19].

The objects in the world may exhibit a feature whose possible values are difficult to distinguish. This occurs when a perceived feature does not have much variability in its range of values: all the values are clustered closely together making it hard to tell the difference between one value and the next.[1] This increases the likelihood of confusion because the usefulness of specifying the feature to a non-expert is diminished (especially if the speaker is more expert than the listener in distinguishing feature values). Hence, if one of these difficult feature values appears in the speaker's

[1] For example, certain Eskimo languages have names for several different states of snow that may be difficult for most non-Eskimos to distinguish [29].

description, the listener, if he isn't an expert, will often relax the feature value to any of the members of the set of feature values.

10.5.4 Relaxing a Description Using Perceptual Knowledge

When examining the features presented in a speaker's description, one can consider perceptual aspects to determine which features are most likely in error. Such an inspection can generate a partial ordering of features for use during the repair process to determine which feature in a description to relax. As shown below, the relaxation ordering suggested by the inspection of features interacts with ordering proposals from other knowledge sources.

Active features are ones that require a listener to do more than simply recognize that a particular feature value belongs to a set of possible values - the listener must perform some kind of evaluation. When considering the water pump domain, it seems that one should first relax those features that require less active consideration such as color (though it is easier to relax red to orange than red to blue), composition, transparency, shape and function. Only after this should one relax those features that require active consideration of the object under discussion and its surroundings (such as superlatives, comparatives, and relative values like size, length, height, thickness, position, distance and weight). People tend to be casual with less active features while the active ones require their full attention. Hence, in a reference failure the source of the problem is likely to be the less active ones.

10.5.5 Discourse Knowledge in Reference

Discourse knowledge concerns discourse structure, the flow of discourse and the use of discourse to highlight parts of the Real World [14, 22, 26, 23, 3, 18]. There are several mechanisms that can highlight objects in discourse (see work on focus by Grosz [14], Reichman [22] and Sidner [26]). This provides a partition of the Real World that prunes down the set of objects to consider during referent identification. When no referent corresponds to a description and recovery is attempted, discourse knowledge can be used to determine whether the problem resides not in the description itself but possibly at the discourse level. For example, midstream corrections in a description by a speaker could confuse a listener causing one to either miss a shift in focus or to shift focus when no shift was intended. The work of [14, 22, 28, 26, 15, 23] provided rules on deictics, anaphoric definite noun phrases, the use of pronominals versus nonpronominals, and so forth, that can be used to clue in on discourse problems.

10.5.6 Hierarchical Knowledge in Reference

Imprecision in a speaker's description can lead to confusion. Being too specific can lead to similar results. Hierarchical knowledge - that is knowledge about hierarchies - can be used by a listener to determine the degree of imprecision or specificity of a description. This effort entails consulting a prestored generic/specific hierarchy of world elements and using the current context to guide the comparison of the current description to elements in the hierarchy.

An imprecise description, missing details to fully distinguish a Real World object, should point out numerous candidates that exhibit the general features in the description rather than none at all. Imprecise descriptions can, however, lead to confusion that blocks the listener from finding a referent. If a feature is difficult to apply because it isn't specific or well-defined, then it may be necessary to ignore it (e.g., the use of a value like "funny" such as in "that funny red thing"). If a feature is ambiguous with respect to how it should be applied, then it may either require relaxation or further restriction (e.g., for the use of a feature value like "rounded," we must ask whether we mean "2-D" or "3-D" rounded? "cylindrical" or "bell-shaped"? and so on). The determination that a feature is too imprecise might be possible before a search for a referent is commenced. An examination of how high in the hierarchy the feature value appears could signal when a more detailed value is needed.

The condition of being too specific is more difficult to detect. In a task-oriented environment, one would not easily notice that something was too specific since normally being very specific is a wise goal for a speaker. The condition of being too specific has drawbacks that occur due to detrimental side-effects. A description can be overspecific if it contains too many feature values. The listener can lose confidence in the referent he found because while it was described with many features, it was very easy to find (with just a few of the features); and so the listener concludes that perhaps he has missed something in his search. Another possible side-effect of being too specific is that the listener ignores the speaker's full description because one feature is too overpowering. That feature dominates the description causing the other features not to be attended to.

10.5.7 Relaxing a Description Using Hierarchical Knowledge

Hierarchical knowledge can resolve certain ambiguities by climbing or descending the hierarchy. This requires looking at a description at two levels: (1) the description's placement in the generic/specific hierarchy and (2) the placement of the filler of each feature of the description in the generic/specific hierarchy.

Hierarchical knowledge also interacts with perceptual knowledge. Confusion can occur when a feature value is too hard to judge. For example, it is difficult to determine which particular feature value applies when the set of possible feature values are too specific. If a more imprecise value is used (and it applies only to one object), it might be easier to find the described object (e.g., "hippopotamus shaped valve" would be better stated as "rounded valve").

10.5.8 Trial and Error Knowledge in Reference

The primary use of trial and error knowledge is to determine whether a referent was properly identified (including ones found with the relaxation process). Performance of a requested action is the strongest determining factor of whether or not the listener correctly interpreted a speaker's description.[1] Successful completion of an action will likely build confidence in the listener that he correctly interpreted a description. Failure to find

[1] In more complex domains - such as ones requiring tools - the actions themselves may be helpful in both finding the referent and confirming whether the choice was correct.

an object after relaxation leads the listener to ask the speaker to clarify; failure to successfully perform the requested action on the object found during referent identification causes the listener to ask himself what is wrong. The trouble might be due to: (1) the object identified from the speaker's description, (2) the action attempted, or (3) some prior (probably unnoticed) mistake that occurred. Failure may come not only from the inability to perform an action but due to an action's postcondition failing.[1] Determination of how badly a postcondition must fail before the listener asks for clarification - instead of reconsidering the description - is unclear and not being investigated here. I will not discuss the actual error recovery that occurs when an action fails since I am currently exploring that area.

10.6 THE RELAXATION COMPONENT

I have discussed some of the numerous kinds of knowledge available to a listener to interpret a speaker's description. I pointed out places where that knowledge affects the listener's ability to interpret a description and ways in which it is helpful to the listener for overcoming poor descriptions. When a description fails to denote properly a referent in the Real World, it is possible to repair it by a relaxation process that ignores or modifies parts of the description. Since a description can specify many features of an

[1] This postcondition need not always be specified explicitly since some postconditions automatically come with an action. For example, if the speaker said the utterance "fit the red gizmo into the bottom side outlet of the main tube," the listener would expect that the red gizmo would fit snugly into the outlet. If, however, it fit loosely, than the listener may feel a mistake has occurred.

object, the order in which parts of it are relaxed is crucial. There are several kinds of relaxation possible. One can ignore a constituent, replace it with something close, replace it with a related value, or change focus (i.e., consider a different group of objects.). In this section, I will describe the overall relaxation component that draws on the knowledge sources as it tries to relax an errorful description to one that suffices.

10.6.1 Find a Referent Using a Reference Mechanism

Identifying the referent of a description entails finding an element in the world that corresponds to the speaker's description (where "described by" means every feature specified in the description is present in the element in the world but not necessarily vice versa). The initial task of our reference mechanism is to determine whether or not a search of the (taxonomic) knowledge base[1] is necessary. A number of aspects of discourse pragmatics can be used in that determination but I will not examine them here.

Assuming that a search of the knowledge base is considered necessary, then the reference search mechanism is invoked. The search mechanism uses the KL-ONE Classifier [17] to search[2] the knowledge base taxonomy. The Classifier uses the subsumption relationships inherent in the taxonomy to place the description in the correct position [17]. What this means with respect to reference is that the possible referents of the description will be found

[1] The knowledge base contains linguistic descriptions and a description of the listener's visual scene itself. Here it is represented in KL-ONE [5], a system for describing inheritance taxonomies.

[2] This search is constrained by a focus mechanism [14, 22, 26].

below the description after it has been classified into the knowledge base taxonomy. If more than one referent is below the classified description, then, unless a quantifier in the description specified more than one element, the speaker's description is ambiguous. If one description is below it, then the intended referent is assumed to have been found. Finally, if no referent is found below the classified description, the relaxation component is invoked.

10.6.2 Collect Votes For or Against Relaxing the Description

It is necessary to determine whether or not the lack of a referent for a description has to do with the description itself (i.e., reference failure) or outside forces that are causing reference confusion.[1] Pragmatic rules are invoked to decide whether or not the description should be relaxed. These rules will not be discussed here.

10.6.3 Perform the Relaxation of the Description

If relaxation is demanded, then the system must (1) find potential referent candidates, (2) determine which features to relax and in what order, and use that to order the potential candidates with respect to the preferred ordering of features, and (3) determine the proper relaxation techniques to use and apply them to the description.

[1] For example, the problem may be with the flow of the conversation and the speaker's and listener's perspectives on it; it may be due to incorrect attachment of a modifier; it may be due to the action requested; and so on.

10.6.3.1 Find potential referent candidates

Before relaxation can take place, potential candidates for referents (which denote elements in the listener's visual scene) must first be found. These candidates are discovered by performing a "walk" in the knowledge base taxonomy in the general vicinity of the speaker's classified description. A scoring KL-ONE partial matcher is used to determine how close candidate descriptions found during the walk are to the speaker's description. The partial matcher generates a score to represent how well the descriptions match (it also generates scores at the feature level to help determine how the features are to be aligned and how well they match). The best of the descriptions returned by the matcher are selected as referent candidates.

10.6.3.2 Order the features and candidates for relaxation

At this point the reference system inspects the speaker's description and the candidates and decides which features to relax and in what order.[1] Once the feature order is created, it determines the order in which to try relaxing the candidates.

Various knowledge sources are consulted to determine the relaxation ordering. These include the perceptual and linguistic knowledge sources that were discussed above, as well as others not discussed in detail here The suggestions from the knowledge sources are then integrated. This integration requires evaluating the partial orderings imposed by each knowledge source.

[1] Of course, once a particular candidate is selected, then deciding which features to relax is relatively trivial - one simply compares feature by feature between the candidate description (the target) and the speaker's description (the pattern).

For example, perceptual knowledge may say to relax color. However, if the color value was asserted in a relative clause, linguistic knowledge would rank color lower. This leads to a conflict. Thus, the relaxation of some other feature may win out over color should it cause less conflict.

Thus, the feature ordering can be used to order candidates: choose first those candidates that best follow the feature order when determining changes that must be made to the speaker's description. The control structure to enforce this rule examines each candidate and assigns a higher priority to those candidates that exhibit a feature ranked higher in the order of features. Hence, the candidates with the least important features slip to the back of the queue.

Once a potential candidate is selected by the controller, the relaxation mechanism begins step 3 of relaxation; it tries to find proper relaxation methods to relax the features that have just been ordered (success in finding such methods "justifies" relaxing the description).

[illegible]3.3.3 Determine which relaxation methods to apply

Relaxation can take place with many aspects of a speaker's description: with the focus of attention in the Real World where one attempts to find a match, with complex relations specified in the description and with individual features of a referent specified by the description.

Often the objects in focus in the Real World implicitly cause other objects to be in focus [14, 28]. The subparts of an object in focus, for example, are reasonable candidates for the referent of a failing description and should be checked. At other times, the speaker might attribute features of a subpart of an object to the whole object (e.g., describing a plunger that is composed of a red handle, a metal rod, a blue cap, and a green cup as "the green plunger"). In these cases, the relaxation mechanism utilizes the part-whole relation.

Complex relations specified in a speaker's description can also be relaxed. These relations include spatial relations (e.g., "the outlet *near* the *top* of the tube"), comparatives (e.g., "the *larger* tube") and superlatives (e.g., "the *longest* tube").

Finally, the simpler features of an object (such as size or color) that are specified in the speaker's description are open to relaxation.

Relaxation of a description has a few global strategies that can be followed: (1) drop the errorful feature value from the description altogether, (2) weaken or tighten the feature value but keep its new value close to the specified one, or (3) try some other feature value.

The realization of these strategies is through a set of procedures (or *relaxation methods*) that are organized hierarchically. Each procedure is an expert at relaxing its particular type of feature. For example, the Generate-Similar-Feature-Values procedure is composed of procedures like Generate-Similar-Shape-Values and Generate-Similar-Size-Values. Each of those procedures are further divided into specialists that first attempt to relax the feature value to one "near" the current one (e.g., one would prefer to first relax the color "red" to "pink" before relaxing it to "blue") and then, if that fails, to try relaxing it to any of the other possible values. If those fail, the feature could be dropped out of consideration.

10.7 CONCLUSIONS

Natural language interactions in the Real World invite contextually poor descriptions. This paper sketches the ideas behind an on-going effort to develop a reference identification mechanism that can exhibit more "human" tolerance of such descriptions. My goal is to build a more robust system that can handle errorful descriptions when looking for a referent, and that is adaptable to existing systems. My work tackles the use of descriptions referring to objects in the Real World and the repair of problems in those descriptions.

The work attempts to provide a computational scheme for handling noun phrases (following the work on noun phrases by [14, 28, 22, 26]) that is robust enough to provide human-like performance. When people are asked to identify objects, they go about it in a certain way: find candidates, adjust as necessary, re-try, and, if necessary, give up and ask for help. I claim that relaxation is an integral part of this process and that the particular parameters of relaxation differ from task to task and person to person. My work provides a forum for trying out the different parameters.

ACKNOWLEDGEMENTS

I want to especially thank Candy Sidner for her insightful comments and suggestions during the course of this work. I'd also like to acknowledge the helpful comments of Chip Bruce, Jeff Gibbons, Diane Litman, Jim Schmolze, Marc Vilain and Dave Waltz on portions of this paper. Many thanks also to Phil Cohen, Scott Fertig and Kathy Starr for providing me with their water pump dialogues and for their invaluable observations on them.

REFERENCES

[1] Agin, Gerald J. Hierarchical Representation of Three-Dimensional Objects Using Verbal Models. Technical Note 182, SRI Internationel, March, 1979.

[2] Allen, James F. *A Plan-Based Approach to Speech Act Recognition*. Ph.D. Th., University of Toronto, 1979.

[3] Allen, James F., Alan M. Frisch, and Diane J. Litman. ARGOT: The Rochester Dialogue System. Proceedings of AAAI-82, Pittsburgh Pa., August, 1982, pp. 66-70.

[4] Appelt, Douglas E. *Planning Natural Language Utterances to Satisfy Multiple Goals*. Ph.D. Th., Stanford University, 1981.

[5] Brachman, Ronald J. *A Structural Paradigm for Representing Knowledge*. Ph.D. Th., Harvard University, 1977.

[6] Brown, John Seely and Kurt VanLehn. "Repair Theory: A Generative Theory of Bugs in Procedural Skills." *Cognitive Science 4*, 4 (1980), 379-426.

[7] Clark, H. H. and C. Marshall. Definite reference and mutual knowledge. In *Elements of Discourse Understanding*, Joshi, Webber and Sags, Ed.,Cambridge University Press, 1981, pp. 10-64.

[8] Cohen, Philip R. *On Knowing What to Say: Planning Speech Acts*. Ph.D. Th., University of Toronto, 1978.

[9] Cohen, Philip R. The need for Referent Identification as a Planned Action. Proceedings of IJCAI-81, Vancouver, B.C., Canada, August, 1981, pp. 31-35.

[10] Cohen, Philip R., Scott Fertig and Kathy Starr. Dependencies of Discourse Structure on the Modality of Communication: Telephone vs. Teletype. Proceedings of ACL, Toronto, Ont., Canada, June, 1982, pp. 28-35.

[11] Goodman, Bradley A. Miscommunication in Task-Oriented Dialogues. KRNL Group Working Paper, Bolt Beranek and Newman Inc., April 1982.

[12] Goodman, Bradley A. Miscommunication and Reference. KRNL Group Working Paper, Bolt Beranek and Newman Inc., January 1983.

[13] Goodman, Bradley A. Repairing Miscommunication: Relaxation in Reference. Proceedings of AAAI-83, Washington, D.C., August, 1983, pp. 134-138.

[14] Grosz, Barbara J. *The Representation and Use of Focus in Dialogue Understanding*. Ph.D. Th., University of California, Berkeley, 1977.

[15] Grosz, Barbara J. Focusing and descriptions in natural language dialogues. In *Elements of Discourse Understanding*, Joshi, Webber and Sags, Ed.,Cambridge University Press, 1981, pp. 84-105.

[16] Joshi, Aravind K. Mutual Beliefs in Question-Answer Systems. In *Mutual Beliefs*, N. Smith, Ed.,Academic Press, 1982, pp. 181-197.

[17] Lipkis, Thomas. A KL-ONE Classifier. Proceedings of the 1981 KL-One Workshop, June, 1982, pp. 128-145.

[18] Litman, Diane. Discourse and Problem Solving. Report No. 5338, Bolt Beranek and Newman Inc., August, 1982.

[19] McDonald, David D. and E. Jeffery Conklin. Salience as a Simplifying Metaphor for Natural Language Generation. Proceedings of AAAI-82, Pittsburgh, Pa., August, 1982, pp. 75-78.

[20] Nadathur, Gopalan and Aravind K. Joshi. Mutual Beliefs in Conversational Systems: Their Role in Referring Expressions. Proceedings of IJCAI-83, Karlsruhe, West Germany, August, 1983, pp. 603-605.

[21] Perrault, C. Raymond and Philip R. Cohen. It's for your own good: a note on inaccurate reference. In *Elements of Discourse Understanding*, Joshi, Webber and Sags, Ed.,Cambridge University Press, 1981, pp. 217-230.

[22] Reichman, Rachel. "Conversational Coherency." *Cognitive Science 2*, 4 (1978), 263-327.

[23] Reichman, Rachel. *Plain Speaking: A Theory and Grammar of Spontaneous Discourse*. Ph.D. Th., Harvard University, 1981.

[24] Ringle, Martin and Bertram Bruce. Conversation Failure. In *Knowledge Representation and Natural Language Processing*, W. Lehnart and M. Ringle, Ed.,Lawrence Erlbaum Associates, 1981.

[25] John R. Searle. *Speech Acts*. Cambridge University Press, 1969.

[26] Sidner, Candace Lee. *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse*. Ph.D. Th., Massachusetts Institute of Technology, 1979.

[27] Sidner, C. L., M. Bates, R. J. Bobrow, R. J. Brachman, P. R. Cohen, D. J. Israel, J. Schmolze, B. L. Webber, W. A. Woods. Research in Knowledge Representation for Natural Language Understanding. Report No. 4785, Bolt Beranek and Newman Inc., November, 1981.

[28] Webber, Bonnie Lynn. *A Formal Approach to Discourse Anaphora*. Ph.D. Th., Harvard University, 1978.

[29] Benjamin Lee Whorf. *Language, Thought, and Reality*. The M.I.T. Press, 1956.

11. PUBLICATIONS

To conclude this report, we present a list a publications by the members of the research group. Also included is a list of presentations by group members.

Publications

Bates, M., and Bobrow, R.J. Natural Language Interfaces: What's Here, What's Coming, and Who Needs It. NYU Symposium on Artificial Intelligence Applications for Business, New York University Graduate School of Business Administration, May 18-20, 1983.

Bates, M., and Bobrow, R.J. Information Retrieval Using a Transportable Natural Language Interface. In *Proceedings of the Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Vol. 17, No. 4, Summer 1983.

Goodman, B.A. Repairing Miscommunication: Relaxation in Reference. In *Proceedings of the National Conference on Artificial Intelligence*, August 22-26, 1983, pp. 134-138.

Heas, A.R. The Syntactic Theory of Belief and Knowledge. Report No. 5368, Bolt Beranek and Newman Inc., Cambridge, MA, September 1983.

Israel, D.J. A Prolegomenon to Situation Semantics. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, June 1983. Also Technical Report No. 5389, Bolt Beranek and Newman Inc., Cambridge, MA, July 1983.

Litman, D. Discourse and Problem Solving. Report No. 5338, Bolt Beranek and Newman Inc., Cambridge, MA, July 1983.

Schmolze, J.G. and Lipkis, T.A. Classification in the KL-ONE Knowledge Representation System. In *Proceedings of the 8th Int'l. Joint Conference on Artificial Intelligence*, August 8-12, 1983, Karlsruhe, West Germany.

Sidner, C.L., and Bates, M. Requirements for National Language Understanding in a System with Graphic Displays. Report No. 5242, Bolt Beranek and Newman Inc., Cambridge, MA, March 1983.

Woods, W.A. A Systems Methodology for Basic Research in Language Use. Paper written for System Development Foundation, Palo Alto, CA, February 1983.

Woods, W.A. Under What Conditions Can a Machine Use Symbols with Meaning? In *Proceedings of the 8th Int'l. Joint Conference on Artificial Intelligence*, August 8-12, 1983, Karlsruhe, West Germany.

Presentations

Bates, M., and Bobrow, R.J. Demonstration of IRUS (Information Retrieval Using the RUS Parsing System). Conference on Applied Natural Language Processing, Santa Monica, CA, February 1983.

Goodman, B.A. Repairing Miscommunication: Relaxation in Reference. Talk given at the National Conference on Artificial Intelligence, August 24, 1983.

Israel, D.J. A Prolegomenon to Situation Semantics. Invited lecture delivered at 21st Annual Conference of the Association for Computational Linguistics, Cambridge, MA, June 1983.

Sidner, C.L. A Computational View of the Pragmatics of Noun Phrases. Invited lecture at the Annual Meeting of the Cognitive Science Society, University of Rochester, NY, May 19, 1983.

Schmolze, J. "Classification in the KL-ONE Knowledge Representation System." Talk given at the 8th Int'l. Joint Conference on Artificial Intelligence, August 8-12, 1983, Karlsruhe, West Germany.

Woods, W.A. Two lectures on Speech Understanding, at SERC/CREST Conference on Speech Processing, University of Cambridge, England. July 1983.

Woods, W.A. "What is Required for a Computer to Use Symbols with Meaning." Talk given at the Int'l. Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, August 1983.

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

cont'd.

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 Ctr.
Computation & Mathematics Dept.
Bethesda, MD 20084

Capt. Grace M. Hopper, USNR 1
Naval Data Automation Command
Code 00H
Washington Navy Yard
Washington, D.C. 20374

Mr. Paul M. Robinson, Jr. 1
NAVDAC 33
Washington Navy Yard
Washington, D.C. 20374

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

Capt. Richard L. Martin, USN 1
507 Breezy Point Crescent
Norfolk, VA 23511

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

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

Major James R. Kreer 1
Chief, Information Sciences
Dept. of the Air Force
Air Force Office of Scientific Research
European Office of Aerospace
Research & Development
Box 14
FPO New York 09510

cont'd.

Mr. Fred M. Griffee
Technical Advisor C3 Division
Marine Corps Development
& Education Command
Quantico, VA 22134 1