COGNITIVE SCIENCE 9, 171-216 (1985)

An Overview of the KL-ONE Knowledge Representation System*

RONALD J. BRACHMAN
Schlumberger Palo Alto Research
JAMES G. SCHMOLZE
BBN Laboratories Inc.

KL-ONE is a system for representing knowledge in Artificial Intelligence programs. It has been developed and refined over a long period and has been used in both basic research and implemented knowledge-based systems in a number of places in the AI community. Here we present the kernel ideas of KL-ONE, emphasizing its ability to form complex structured descriptions. In addition to detailing all of KL-ONE's description-forming structures, we discuss a bit of the philosophy underlying the system, highlight notions of taxonomy and classification that are central to it, and include an extended example of the use of KL-ONE and its classifier in a recognition task.

1. INTRODUCTION

KL-ONE is a system for representing knowledge in Artificial Intelligence programs, more or less in the tradition of semantic networks and frames.

* The history of the ideas in KL-ONE is too complex to recount here, and the number of contributors to this whole area of work makes it inevitable that we will forget to mention many who deserve to be acknowledged. At the very least, we would like to extend our thanks to these people, who have been intimately involved with KL-ONE, KL-TWO, and Krypton: Danny Bobrow, Rusty Bobrow, Phil Cohen, Richard Fikes, Mike Freeman, Jeff Gibbons, Victoria Gilbert, Brad Goodman, Norton Greenfeld, Austin Henderson, David Israel, Henry Leitner, Hector Levesque, Bob Lingard, Tom Lipkis, Bill Mark, Peter Patel-Schneider, Candy Sidner, Mark Stefik, Marc Vilain, David Wilczynski, Bill Woods, Martin Yonke, and Frank Zdybel.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-77-C-0378. Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

Correspondence and requests for reprints should be sent to Ronald J. Brachman, AT&T, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974.

Literally speaking, KL-ONE (a k a KLONE) is an implementation of some ideas about the structure of descriptions and their use in reasoning, a computational incarnation of what have been called *structured inheritance networks* (or *SI-Nets,* see Brachman, 1978, in press, b).[1] But its utility has gone well beyond that of an implementation.

KL-ONE first appeared in 1977 as the initial implementation of a representational paradigm described in the first author's Ph.D. dissertation (Brachman, 1978, in press, b). The original work developed a level of representation that was independent of any particular domain, but whose primitives were more explicitly geared to the task of AI knowledge representation than those of predicate logic.[2] This level of representation—for better or worse called the "epistemological level"—tried to deal carefully with ideas of "description," "attribute," "concept," "role," "inheritance," and "instantiation," which were treated in a somewhat informal manner in previous representation systems. In that regard, the work owes much to Woods (1975) and Brachman (1977), which criticized various inconsistencies and ambiguities in semantic network systems.[3]

KL-ONE was originally used in two systems at Bolt Beranek and Newman Inc.: a system for intelligent information presentation (Zdybel, Greenfeld, Yonke, & Gibbons, 1981) and a large prototype natural language understanding system (Brachman et al., 1979; Sidner, Bates, Bobrow, Brachman, Cohen, Webber, & Woods, 1981). In these two contexts, KL-ONE provided a useful set of primitives for forming descriptions of the objects of the domain, as well as an inference mechanism for deriving the consequences of the use of descriptions in particular situations (see the Appendix for an example of KL-ONE's utility in the natural language system). Since then, the representation system has grown, its representational facilities have matured, and its user community has expanded. It has inspired several new research efforts.[4] Its research community has had three workshops (see, for instance, Schmolze & Brachman, 1982), and KL-ONE has

[1] KL-ONE has as an integral part a language for specifying descriptions and occasionally the name has been used to refer to just that language. However, the implementation comprises much more than the language; it includes facilities for building and saving KL-ONE networks, querying a network, etc. That is why "system" is probably the most appropriate descriptor for KL-ONE.

[2] The languages of mathematical logic were developed for the precise expression of mathematical propositions and the generation of their consequences using combinatorial rules of proof. See Israel (1984), especially section 4.2, for a detailed discussion on the appropriateness of logic as an AI knowledge representation language. Also, see Brachman, Fikes, and Levesque (1983) for a discussion of the role of logic vis-a-vis AI description languages.

[3] For an introduction to semantic networks, and a more detailed justification of the kind of representation exemplified by KL-ONE, see Brachman (1979). A brief recap is presented in section 2.1 of this paper.

[4] The first author is now working exclusively on one of these (Brachman et al., 1983), and the second author is working on a new implementation of KL-ONE (Moser, 1983).

been used in systems for understanding and generating natural language, interactive information retrieval, question-answering about system utilities and natural language command execution, computer system configuration, and office procedures modeling. KL-ONE has also influenced work in philosophy and psychology (Cohen, 1982; Rifkin, 1985).

With all of this, KL-ONE has achieved a status afforded to few efforts in the brief history of Artificial Intelligence: Over a significant period of time (at least eight years) and over a large number of projects, it has served as the foundation for some very basic research in knowledge representation (Brachman, 1983; Brachman, in press, a; Israel, 1983; Israel & Brachman, 1984), and at the same time has provided representational support in a number of implemented AI systems. Over its history, the language of KL-ONE has of course changed. However, throughout its many implementations (at least three in Interlisp as well as versions in SmallTalk (Fikes, 1982), PROLOG (Freeman, Hirschman, McKay, Miller, & Sidhu, 1983), SNePs (Tranchell, 1982), and GRASPER (Woolf, 1982), KL-ONE has maintained an unchanging central core of ideas and representational philosophy. It is probably this representational kernel that is responsible for the interest in, and longevity and utility of, KL-ONE, and it is this kernel of ideas to which we address ourselves in this paper.

In the KL-ONE kernel discussed here, we concentrate heavily on the static structure of and interrelations between descriptions. A great deal of the recent work on knowledge representation in AI has concentrated on the forms of representations, and the work on KL-ONE is no exception. Unfortunately, the representation forms developed do not wear their applicability on their sleeves (this is just as true of logic and English as representation languages as it is of KL-ONE or KRL). Throughout the text, we do mention inferences that follow from structuring a domain in a KL-ONE knowledge base and such automatic deductions are a central benefit of representing knowledge with KL-ONE. The only place, however, that we try to address a seriously complicated use of KL-ONE is in the Appendix.

2. LANGUAGE STRUCTURE AND PHILSOPHY

KL-ONE principally provides a language for the explicit representation of conceptual information based on the idea of *structured inheritance networks* (Brachman, 1978, 1979, in press, b). Before going into the details of KL-ONE structures, we will first sketch the philosophy underlying the development of the language.

KL-ONE is intended to represent general conceptual information and is typically used in the construction of the knowledge base of a single reasoning entity. A KL-ONE knowledge base can be thought of as representing the

beliefs of the system using it. Thus KL-ONE fits squarely into the currently prevailing philosophy for building reasoning systems. This approach to knowledge-based systems is characterized by what Brian Smith (1982) calls the "knowledge representation hypothesis":

> Any mechanically embodied intelligent process will be comprised of structural ingredients that (a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and (b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge. (p. 2)

In other words, KL-ONE provides a language for expressing an *explicit set of beliefs* for a rational agent.[5]

KL-ONE aspires to a bipartite view of the knowledge-representation task. Over the course of its development, we began to tease out the distinction between KL-ONE constructs whose intent was primarily for elaborating descriptions and those whose intent was for making statements. In a sense, KL-ONE was beginning to divide into two different formalisms—one for *assertion* and one for *description*. These two parts would serve to represent the beliefs of the system and the terms out of which the belief sentences would be constructed, respectively.[6]

While KL-ONE never really split into formally distinct sublanguages, we often speak as if it had. The intent behind the different kinds of constructs is quite different. In particular, the descriptional part of KL-ONE allows one to form a variety of descriptive terms out of other descriptive terms using a small set of description-forming operators. This yields an extensible repertoire of terms—a conceptual vocabulary—that can be used to make assertions. For example, we can form a compound KL-ONE description corresponding to "a man from Betelgeuse" using the KL-ONE descriptions for "a man" and "Betelgeuse." However, simply forming this description asserts nothing about any particular man or star, as structures in the description language have no assertional import by themselves (but see section 4). The assertional part, on the other hand, makes use of terms from the description language to make statements about the world. The assertional capabilities in KL-ONE are somewhat impoverished as compared, say, to a first order language with equality; they include only statements of existence, of coreference of description, and of identity of individual constants (all in a particular context). For example, we might form a description like "the person giving the talk" and use it to assert that such a person exists (in say, context C1). We could then establish another description—say, "a man from Betelgeuse"—as coreferential with the first and thereby make the

[5] Such a set of beliefs expressed in some representation language is what is typically meant by the term *knowledge base*.

[6] This line of thought is more developed in Brachman et al. (1983); the separation between definition and assertion is one of the *raisons d'être* of the Krypton system.

statement (in C1), "the person giving the talk is a man from Betelgeuse." This latter type of construct—a predication—is a legitimate object of belief, whereas descriptions by themselves are not.

This paper describes the KL-ONE language and system as of the summer of 1982, which marks the end of a chapter of our work in knowledge representation. Until then, most of our work on KL-ONE had focused on description formation, with very little attention paid to making assertions (we felt that existing formalisms, such as predicate logic, were adequate for this task). Since that time, our thoughts about description formation have extended the work described herein, and we have begun to focus more intensively on the assertion language. This work has been taken up in the context of two new experimental representation systems, Krypton (Brachman et al., 1983) and KL-TWO (Moser, 1983).

2.1 Epistemological Primitives

KL-ONE is, in a sense, an "object-centered" language. Its development has proceeded from traditional semantic networks, but its principal structures do not directly represent either propositions or sets as did those of several earlier semantic net systems (e.g., see Hendrix, 1979, and Schubert, Goebel, & Cercone, 1979).[7] Instead, the principal element of KL-ONE is the *structured conceptual object,* or *Concept.*[8]

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

Bearing in mind the value of consistency at a single level of network primitive, we have set out to capture an adequate set of primitive elements for representing a broad spectrum of concepts. We have attempted to deter-

[7] "Object-centeredness" is a characteristic of many of the current representation systems. Its prominence seems to arise from the convenience of indexing knowledge through the entities that the knowledge is about, and the whole area of "object-oriented programming" has grown up in parallel. See Brachman (in press, section 4.2) and Nilsson (1980, section 9.1), for discussions of this feature.

[8] KL-ONE object types will be capitalized throughout this article.

mine a reasonable set of underlying object and relation types for knowledge structuring. To the extent that we can formalize this in a grammar for well-formed conceptual structures, we have defined what might be called an "epistemology." This is not a theory of any particular domain—one builds that on top of this level—but part of a generative theory of the structure and limits of thought for a rational agent.[9] KL-ONE thus comprises a fixed set of "epistemologically primitive" structure types (e.g., "Concept," "Role") and structure-forming operations (e.g., "specialization," "restriction," "differentiation"). We have attempted to understand the important features of the internal structure of concepts, and to embody them in a language that is expressively powerful and fairly natural to use.

2.2 Primitive and Defined Concepts

KL-ONE separates its descriptions into two basic groups: *primitive*[10] and *defined*. When specifying domain knowledge in KL-ONE, one usually first specifies some primitive types, which are then typically followed by other types (either primitive or defined) that are specified in terms of them.

For example, if our domain were planar geometry, we might begin with POINT and LINE SEGMENT as atomic, primitive types; these would be represented in KL-ONE by *primitive Concepts*. We might also decide that the concept of a polygon was useful to represent, but while we had several necessary properties of polygons in hand, we did not want to attempt to characterize fully its necessary and sufficient conditions. In this case, POLYGON would also have to be a primitive Concept. However, KL-ONE allows primitive Concepts to have defined properties; that is, it treats primitive Concepts as incomplete definitions. This means that we could include in its specification the fact that a polygon—by definition—has three or more sides that are line segments. (Nonatomic but still primitive concepts are also discussed in Israel, 1983.)

Once given the POLYGON Concept, we could specify TRIANGLE as a *defined Concept* derived from it. Namely, a triangle is exactly a polygon

[9] KL-ONE does not commit one to any particular domain primitives but rather provides a representational foundation out of which domain primitives can be specified. Generally speaking, a representer ("knowledge engineer" in some circles) selects his domain primitives with a particular set of useful inferences in mind. See Amarel (1968) for an example of how a change in domain-level primitives can help solve a problem.

KL-ONE also takes a strong stand on names. As has been pointed out (for instance, see Brachman, in press, a; Israel & Brachman, 1984; McDermott, 1982; and Woods, 1975), suggestive names can do more harm than good in semantic networks and other representation schemes. Atomic labels attached to nodes in KL-ONE are purely for user convenience and hold no significance for any KL-ONE functions.

[10] "Primitive" here refers to domain concepts for which we are incapable of giving full necessary and sufficient definitions. In the previous subsection, we discussed primitive objects and operations for the KL-ONE language.

with three sides. TRIANGLE becomes a new term in the description language, defined to be nothing more than "a polygon with exactly 3 sides," a definition that gives both necessary and sufficient conditions for being a triangle. Given a plane figure that is at least a polygon and has three sides (it may have other properties), the figure is a triangle simply on the basis of the meaning of the term. On the other hand, given the primitive specification for polygons, even an object satisfying the description would not be guaranteed to be a polygon. For example, a particular geometric figure with three sides that were line segments might not be a polygon—it might not be closed.

Although a primitive Concept does not provide sufficient conditions, it can specify a rich variety of necessary conditions. The notion of *natural kind terms* such as *dog* or *lemon,* may be related to this last point, because it is usually assumed that it is not possible to completely define such terms. Even so, it is probably important to allow elaboration of natural kind terms in KL-ONE, and some of this might be specified with necessary conditions. For example, *mammal* and *cat* are both natural kinds, and there is an important relationship between them that we might want to represent, namely, that cats are necessarily mammals (see Kripke, 1980, esp. pp. 122–128). KL-ONE allows a Concept for cats to be specified that includes that relationship.

Another important type of knowledge about natural kinds and other real-world categories is default or typicality information. KL-ONE as yet does not address this directly, although we do comment on it in section 7.

3. NETWORKS AND THE NOTION OF A CLASSIFIER

As mentioned earlier, KL-ONE is based on the idea of *structured inheritance networks.* What this amounts to is that it is convenient to think of a KL-ONE knowledge base as a type of semantic network with a roughly hierarchical organization of general types (called *Generic Concepts*). The "structured inheritance" aspect refers to the fact that an implementation must preserve a complex set of relations between description parts as one moves down the specialization hierarchy; the details of this will become evident later.

In KL-ONE the network implementation follows from the structure of the description language. That there is a type hierarchy as the backbone of a KL-ONE knowledge base is derived from the fact that KL-ONE descriptions are always formed from other, more general KL-ONE descriptions. The specialization relations implicit in these compound descriptions (e.g., to move rather abruptly to the domain of electronic mail, "message from AAAI-OFFICE@SUMEX" is a compound description that implicitly specializes "message") are naturally envisioned in a directed graph structure. While it is easy to think of KL-ONE structures in terms of nodes and

links, this is only an incidental byproduct of the relations implicit in the language of Concepts and Roles.

Given two KL-ONE descriptions, an important question to consider is whether one *subsumes* the other—that is, whether an instance of one is always an instance of the other. In semantic nets, this question usually comes down to looking from one node up the hierarchy to see if another happens to lie on a superset path. In KL-ONE, the subsumption question can also be answered by looking up a hierarchy, with one crucial difference. Because the network is simply a byproduct of the structure of terms in the language (the network is not itself the language), not all network-derived subsumption inferences are valid unless the hierarchy completely reflects all of the relations implicit in the descriptions in question. In other words, the descriptions must be in their proper places in the network before any conclusions can be drawn.

This gives rise to the notion of a *classifier* (Schmolze & Lipkis, 1983), which is a mechanism for taking a new KL-ONE description and putting it where it belongs in the hierarchy. It is in the right place if it is below all descriptions that subsume it, and if it is above all descriptions that it subsumes.[11] Classification provides an important inference capability to a system using KL-ONE, and a detailed example of its use appears in the Appendix.

All in all, then, KL-ONE knowledge bases have a network flavor, with the links standing for what we have called the "epistemologically primitive" relations among concepts. The network is a reflection of the implicit subsumption (and other) relations among the descriptions that its nodes stand for.

4. GENERIC CONCEPTS AND BASIC TAXONOMY

As mentioned, the principal elements of KL-ONE descriptions are Concepts, of which there are several types. The most important type is the *Generic Concept,* the KL-ONE equivalent of a "general term" (Quine, 1960)—potentially many individuals in any possible world can be described by it. For example, a KL-ONE knowledge base might have Generic Concepts for

[11] There are differences in philosophy on computing subsumption in different KL-ONE-based systems. In Krypton (Brachman et al., 1983), the subsumption relation is computed directly from the descriptions, and is only stored in a network for later computational convenience. In KL-ONE, the network is computed first from the forms of descriptions, and subsumption questions are always read off from the hierarchy. In either case, it is important to have a means of computing subsumption independent from a simple network lookup. This independent means of determining subsumption makes soundness an important property of KL-ONE, as opposed to many other semantic network systems, whose only semantics is "what the interpreter does."

animal, mammal, human, female human, woman, etc., each of which are descriptions that could be used to describe many individuals in the world.

In fact, as already hinted, some of these Generic Concepts may be formed out of the others (the Generic Concept *HUMAN* would have *MAMMAL* as a component, for example[12]). There are several structure-forming operations available for building Concepts,[13] which bring together one or more general Concepts and a set of restrictions on those Concepts. More specifically, the components of a Concept are

- its subsuming Concepts (its *superConcepts*).
- and its local internal structure expressed in
  - *Roles,* which describe potential relationships between instances of the Concept and those of other closely associated Concepts (i.e., its properties, parts, etc.), and
  - *Structural Descriptions,* which express the interrelations among the Roles.

To be well-formed, a KL-ONE Concept must have more than one superConcept (if there are no local restrictions), differ from its superConcept in at least one restriction, or be primitive. A Concept with no local restrictions is defined as the conjunction of its superConcepts.

The Roles and Structural Descriptions of a Concept are taken as a set of restrictions applied to its superConcepts. Thus, a superConcept serves as a proximate genus, whereas the local internal structure expresses essential differences, as in classical classificatory definition (Sellars, 1917). It should be noted that its superConcepts and set of restrictions are the only KL-ONE structures that contribute to the meaning of a Concept.

As mentioned in section 3, when one specifies a Concept like *HUMAN* in terms of one like *MAMMAL,* one is implicitly specifying that the more general subsumes the more specific. Subsumption of descriptions has the following consequence: If Concept *A* subsumes Concept *B,* then every individual that can be described by *B* can also be described by *A*.[14] So, by specifying the Concept *HUMAN* such that the Concept *MAMMAL* subsumes it—in other words, so that the property of being a human includes the property of being a mammal—one is specifying that any human must be a mammal.

[12] We will use upper-case, italic letters when writing the names of Generic Concepts, which KL-ONE allows one to specify for convenience. The names carry no meaning for the system.

[13] From this point on, we will use "Concept" to mean "Generic Concept," except when "Concept" alone would be ambiguous.

[14] More precisely, *A* subsumes *B* if, and only if, in all possible interpretations, the extension of *A* is a superset of the extension *B*. We have specified a formal extensional semantics for at least part of KL-ONE (see Schmolze & Israel, 1983).

We often refer to the network structure formed by the subsumption relationships between Concepts as a "taxonomy." Whereas a Concept like *WOMAN* might be subsumed by all of *THING, ANIMAL, HUMAN,* and *FEMALE-ANIMAL,* the taxonomy usually indicates only the direct subsumption relations. Because this relation is transitive, the relation between *WOMAN* and, say, *ANIMAL* can still be read off of the network. This is typical of semantic network taxonomies, and makes the notation more readable.

For example, the simple taxonomy of Figure 1 shows for each Concept only the proximate genus. Each ellipse represents a Generic Concept; the subsumption relation is denoted by a *superC* link, which is depicted by a wide arrow. The superC link is sometimes called a superC *cable* because, as we will see, other links may be associated with it.

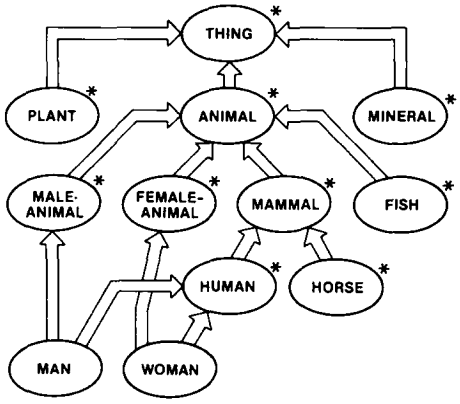


Figure 1. A simple KL-ONE network of Generic Concepts.

In using a compact notation for Concepts, wherein only "local" information is indicated, we must be careful not to neglect certain inferences that are dictated by the meaning of the subsumption relation. First, as mentioned, a Generic Concept actually subsumes all Generic Concepts below it, either immediately, or otherwise.[15] Also, because a Concept is defined in terms of its superConcepts, all of the restrictions (the essential differences) of the parents must apply to the children. In order to achieve this effect, the KL-ONE system provides *inheritance* facilities. Regarding our example, all component restrictions of the Concept *MAMMAL* would be inherited by *HUMAN* (see the sections on Roles and Structural Descriptions for the exact specification of these restrictions). So, if the Concept *MAMMAL* included components that meant that mammals were warm-blooded animals, the Concept *HUMAN* would inherit those same components.

[15] From this point on, the notions of "above" and "below" will be used interchangeably with "subsumer" and "subsumee," respectively.

Figure 1 illustrates some other points about KL-ONE networks. For one thing, KL-ONE taxonomies always have a single root Concept, usually named *THING*. *THING* subsumes all other Concepts and is the only one that has no superConcepts. For another, as we discussed in section 2.2, some Concepts are fully defined by their components and some are not. In Figure 1, the Concepts with an asterisk are primitive, the others are defined.

You may also note in Figure 1 that a Generic Concept can have many superConcepts as well as subConcepts. The Concept *WOMAN* has both *FEMALE-ANIMAL* and *HUMAN* as its superConcepts. Hence, both *FEMALE-ANIMAL* and *HUMAN* subsume *WOMAN*. The Concept *WOMAN* in this case is defined to be just the conjunction of the two.

Finally, it is important to reiterate that a Concept like *MAMMAL* does not derive any of its meaning from the Concept *HUMAN*. A Concept's meaning is strictly determined by its subsuming Concepts plus the information associated specifically with the Concept.[16]

5. ROLES, RESTRICTION, AND DIFFERENTIATION

The Role is the primary component of a Concept. A Role acts like a generalized attribute description, representing potential relationships between individuals of the type denoted by the Concept and other individuals. In other words, Roles are the KL-ONE equivalent of two-place predicates.

KL-ONE distinguishes Roles from their fillers. The difference is motivated essentially by the "attributive/referential" distinction in the philosophy of language (Donnellan, 1966). Imagine a situation in which an alligator's tail has fallen off. We might remark, "The alligator's tail lay wriggling on the ground." Or, we might say something like, "Don't worry, the alligator's tail will grow back again." The "tails" talked about must be different in the two cases—in the first, we are referring to the previous filler, the actual piece of protoplasm that used to be the alligator's tail. In the second, because the previous tail will not reattach itself to the alligator, we must mean something else by "alligator's tail." We are in fact talking in a general way about anything that will eventually play the role of "tail" for the alligator. KL-ONE lets us distinguish statements about an actual known role filler and a potential one by providing an explicit structure for the description of potential fillers, the Role.

[16] There has been considerable discussion among KL-ONE users as to whether or not KL-ONE should be able to represent exhaustion or mutual exclusion among a Concept's subsumees. If it did, then a Concept could possibly gather part of its definition from those it subsumes, in the case of exhaustion, or "sibling" Concepts (Concepts that share the same subsuming Concept), in the case of mutual exclusion. However, KL-ONE does not support either, although KL-TWO (Moser, 1983) does. Krypton (Brachman et al., 1983) allows mutually exclusive terms to be defined.

There are several different types of Roles of which the Generic RoleSet is the most important. RoleSets (in general, used to mean Generic RoleSet) capture the notion that a given functional role of a Concept (e.g., sender of a message, upright of an arch, officer of a company, input to a program) can be played by several different entities for just one individual. A RoleSet captures the commonality among a set of individual role players (e.g., what all officers of a given company will have in common by definition).

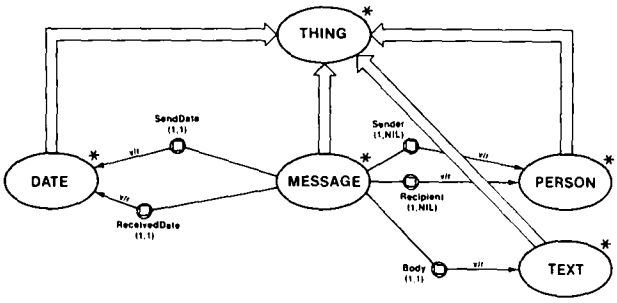
A pictorial representation of a Concept of a message is shown in Figure 2, where *MESSAGE* represents a simple type of electronic message. Here, we see that *MESSAGE* has *THING* as a subsumer (because this is true of all Concepts, *THING* will not appear in any other figures). Each of the encircled squares depicts a Generic RoleSet, of which *MESSAGE* has five. (For convenience, KL-ONE allows Roles to be named, and in this paper we have named all Roles after the relations they represent. Role names appear in the text as italicized, capitalized words.) The RoleSets in the figure are connected to *MESSAGE* by unnamed links that merely denote that the RoleSets are components of *MESSAGE* (we sometimes call the link "has-role").

The quoted sentence at the bottom of Figure 2 is a JARGON statement specifying the Concept. JARGON is a stylized, restricted, Englishlike language for describing KL-ONE objects and relationships. It has two important properties: It is usually easier for a novice to understand a JARGON statement than its equivalent in the graphical notation, and an interpreter exists that can translate most JARGON statements into appropriate KL-ONE structures.[17]

RoleSets themselves have structure. Descriptions of potential fillers are specified with a *Value Restriction (V/R).* In Figure 2 we see that the RoleSet *Sender* has a Value Restriction of *PERSON.* The system interprets Value Restrictions as *necessary* type restrictions on RoleSet fillers, and so the senders of messages must be persons. No cancellation of Value Restrictions is allowed (for example, in this ontology, senders of *any* subtype of message must be persons—see section 7). Cases arise where several Value Restrictions are applicable to a RoleSet filler (these cases will be apparent as the inheritance mechanism is explained). If more than one V/R is applicable at a given RoleSet, the restrictions are taken conjunctively.

Because the functional roles defined by RoleSets can be played by more than one individual at a time, RoleSets also have *Number Restrictions* to express cardinality information. A Number Restriction is a pair of numbers, a lower and upper bound, defining the range of cardinalities for sets of role-player descriptions. We use "NIL" for infinity in cases where there is

[17] We have taken some liberties with JARGON in our figure captions to improve readability. For the complete story on what JARGON actually can do, see Woods (1979).



"A MESSAGE is, among other things, a THING with at least one Sender, all of which are PERSONs, at lease one Recipient, all of which are PERSONs, a Body, which is a TEXT, a SendDate, which is a DATE, and a ReceivedDate, which is a DATE."

Figure 2. The Primitive Concept MESSAGE.

no finite upper bound. Thus the Number Restrictions in Figure 2 (written in parentheses near the RoleSets) indicate that any *MESSAGE* has at least one *Sender,* at least one *Recipient,* exactly one *Body,* exactly one *SendDate,* and exactly one *ReceivedDate.*

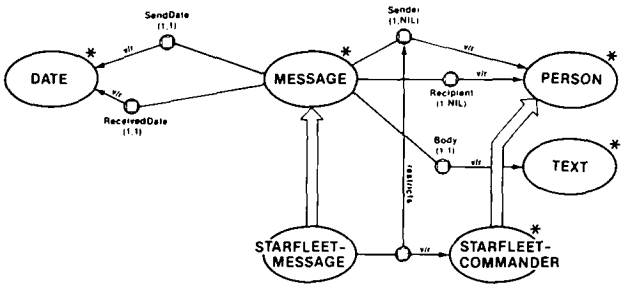
5.1. RoleSet Restriction

In section 4 we defined the subsumption relation between Concepts, which reflects how one Concept can be specified in terms of another. The *restriction* relation[18] between Role Sets allows a similar specification, but with respect to the components of a RoleSet. In Figure 3, we have shown the Concept *STARFLEET-MESSAGE*, which represents messages sent by Starfleet commanders. There is a link labeled "restricts" from the RoleSet on the lower Concept to *Sender,* which means that this lower RoleSet denotes a subset of the relation denoted by *Sender,* and in this case, that subset is restricted to senders who are Starfleet commanders.

We define restriction with the following: If Concept *A* with RoleSet *Ra* subsumes Concept *B,* and if RoleSet *Rb* of *B* restricts *Ra,* then every set of fillers of *Rb* satisfies all restrictions on *both Ra* and *Rb.* Moreover, *Ra* and *Rb* designate the same two-place relation.

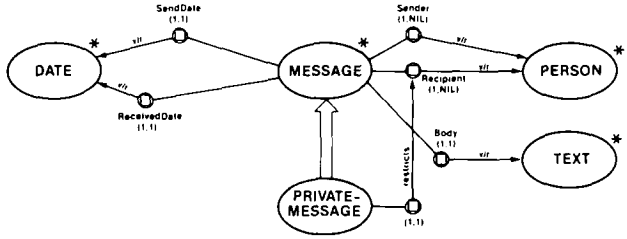
Restriction does not specify a new Role, rather it adds constraints on the fillers of a Role with respect to some Concept. These constraints include those specified by both Value Restrictions and Number Restrictions, such as in Figure 4. For a *MESSAGE* to be a *PRIVATE-MESSAGE,* it must have

[18] For most of KL-ONE's history, this has been called "modification," but "restriction" is more appropriate.



"A STARFLEET-MESSAGE is a MESSAGE, all of whose Senders are STARFLEET-COMMANDERS."

Figure 3. A Defined Concept that uses Role restriction.



"A PRIVATE-MESSAGE is a MESSAGE with exactly one Recipient."

Figure 4. Another Defined Concept that uses restriction.

exactly one *Recipient*. Our graphical notation unfortunately does not distinguish newly introduced RoleSets (such as *Recipient* for *MESSAGE*) from already inherited RoleSets whose components are just being further restricted (such as *Recipient* for *PRIVATE-MESSAGE*). The only way to tell them apart is that the latter have "restricts" links pointing away from them. By convention, restricted RoleSets inherit the names of the RoleSets they restrict, so in Figure 4 the restricting RoleSet inherits the name "Recipient."

The figures in this paper emphasize the local Concept-forming operations and thus do not usually include the inherited components of a Concept. However, it should be kept in mind that the meaning of a Concept includes not only its local restrictions, but its inherited components as well. Just for reference, Figure 5 illustrates the "true" picture of the Concept *PRIVATE-MESSAGE*. It shows the Concept with all of its components. Note that when queried, the KL-ONE implementation provides inherited information about a Concept or Role, thus performing an important kind of inference at retrieval time. If one tried to draw a picture of *PRIVATE-MESSAGE* solely from the results of querying the system about the components of a *PRIVATE-MESSAGE*, the result would be as in Figure 5.

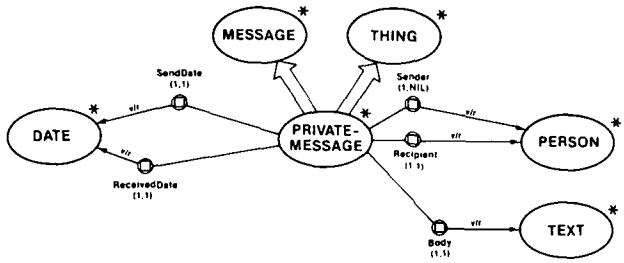


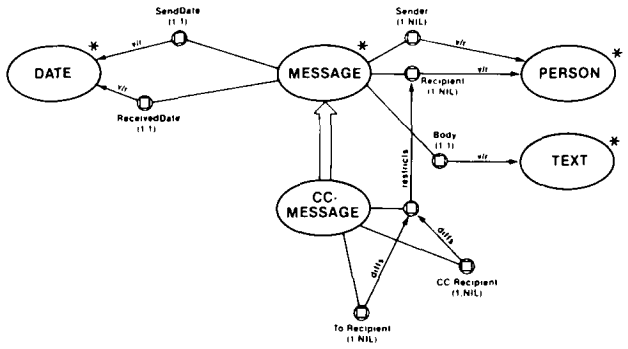
Figure 5. A depiction of all inherited components of a Concept.

5.2. RoleSet Differentiation

RoleSet *differentiation* is one of KL-ONE's unique features. A RoleSet differentiates another when the former denotes a subrelation of the relation denoted by the latter. The motivation for differentiation comes from the fact that KL-ONE Roles are intrinsically set descriptions, with potentially many fillers for a given Role; differentiation allows the specification of sub-Roles that are to be filled with subsets of the fillers of the Roles they differentiate. For example, one might want to differentiate the *Officer* Role of *COMPANY* into *President, Vice President,* etc. KL-ONE allows definitional knowledge common to all types of officers to be associated with the *Officer* Role, and that specific to president, etc., to be associated with the subRoles of *officer*. Further, a differentiation relation between *President* and *Officer* specifies that any president of a company is, by definition, an officer of that company.

Figure 6 demonstrates differentiation with the Concept *CC-MESSAGE,* which splits the *Recipient* RoleSet of *MESSAGE* into two parts, *To-Recipient* and *CC-Recipient*. Many electronic mail facilities allow the separation of recipients of messages into two categories: those to whom the message is primarily addressed (the fillers of the *To-Recipient* RoleSet), and those who should receive the electronic equivalent of a carbon copy (the fillers of the *CC-Recipient* RoleSet). Differentiation lets one specify that the *To-Recipient* is indeed a *Recipient,* and that this is a necessary condition.

The behavior of differentiating RoleSets (e.g., *To-Recipient* in the figure) with respect to Value Restrictions is the same as in the case of Role restriction. A subRoleSet inherits the Value Restriction of the RoleSet it differentiates. When specifying a subRoleSet, one may also specify additional constraints for the Value Restriction. In such cases, both the inherited Value Restriction and the locally specified one must apply (i.e., their conjunction must apply). On the other hand, Number Restriction inheritance works a little differently than in the case of restriction. Because the essence of differentiation is the specification of a subset, only the maximum can be inherited.



"A CC-MESSAGE is a MESSAGE, at least one of whose Recipients is a To-Recipient, and at least one of whose Recipients is a CC-Recipient."

Figure 6. A Concept that uses RoleSet differentiation.

If no minimum is specified at the subRole, the minimum is taken to be 1 (not the minimum of the parent). Note that a differentiating RoleSet also inherits the name(s) of the RoleSet it differentiates. This is approriate since by definition all fillers of the subRoleSet are also fillers of the parent (a to-recipient is also a recipient).

It is important to note that the information associated with a differentiating RoleSet specifies necessary conditions, but not sufficient ones. In this regard, the relation denoted by such RoleSets can be thought of as primitive. For our *CC-MESSAGE* Concept, we have not specified precisely under what conditions a Recipient is either a *To-Recipient* of a *CC-Recipient* or neither.[19]

We can define RoleSet differentiation a little more formally with the following:

> If RoleSet *Rb* differentiates RoleSet *Ra*, then any pair of individuals that satisfy the relation denoted by *Rb* also satisfy the relation denoted by *Ra*. Furthermore, all individuals in the range of the relation denoted by *Rb* satisfy the Value Restrictions of both *Ra* and *Rb*. The maximum (cardinality of the image of the relation for any individual in the domain) specified in the Number Restriction of *Ra* is also the maximum for *Rb*, unless a smaller maximum is specified directly at *Rb*. The minimum for *Rb* is 1, unless a larger minimum is specified directly at *Rb*.

Both restricting RoleSets and differentiating RoleSets can themselves be restricted and/or differentiated (for differentiation, as long as the maxi-

[19] The Structural Description mechanism, which will be covered briefly later, was designed to provide the missing part of the definitions of Roles. While interesting work is proceeding using special cases of SDs (see, for example, Freeman & Tomlinson, 1982; Freeman, Hirschman, McKay, & Palmer, 1983), the details of the Role-defining mechanism have not yet been worked out satisfactorily.

mum is greater than 1) in the specification of other Concepts. Thus, except for the subtlety about Number Restrictions, differentiation between Role-Sets is similar to subsumption between Concepts. KL-ONE actually supports a *Role taxonomy* akin to its basic Concept taxonomy.

At this point, we have described enough of KL-ONE to explain the classifier. We will continue with the remainder of the KL-ONE syntax immediately thereafter.

6. CLASSIFICATION OF KL-ONE CONCEPTS

The classifier takes a newly specified Concept and determines the subsumption relations between it and all other Concepts in a given network. In some cases subsumption is specified directly, as in Figure 3, where *MESSAGE* was specified as *STARFLEET-MESSAGE's* subsumer. Indirectly, this also implies other cases of subsumption due to transitivity—e.g., in the same figure all subsumers of *MESSAGE* also subsume *STARFLEET-MESSAGE* (in this case only *THING*). However, classification also discovers cases of subsumption not readable from the Concept specification by simple means, and in such cases, the classifier adds the appropriate superC links.

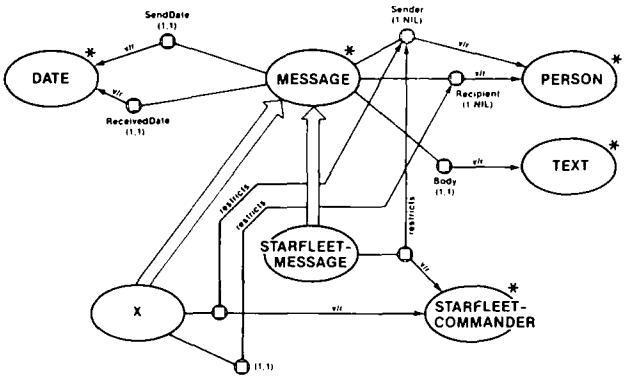
An example of the classifier discovering a subsumption relation is shown in Figures 7 and 8. In Figure 7, we specify Concept *X* as "a *MESSAGE* with exactly one *Recipient* and whose *Sender* is a *STARFLEET-COMMANDER*." This specification does not make transparent the fact that *STARFLEET-MESSAGE* subsumes *X*, but the classifier will discover that relation and add a superC link (as shown in Figure 8).[20]

The classifier takes all components of a Concept's specification into account (only some of which have been described so far). We have shown informally (though not here) that the classifier's algorithm is sound, i.e., any subsumption relations discovered by the classifier are legitimate, but not complete, i.e., it does not discover all subsumption relations.[21] While no formal specification of its incompleteness has been made, the cases missed by the classifier have not proven problematic in applications of KL-ONE to date.

The effect of the classifier is to automate the placement of new Concepts into a KL-ONE taxonomy. The proper place for a Concept is above those Concepts it subsumes (its subsumees) and below those that subsume it (its subsumers). Not only does this simplify the task of creating static knowl-

[20] The classifier will also discover that *X* is a *PRIVATE-MESSAGE*, although this is not shown in Figure 8.

[21] The issues of soundness and completeness for the KL-ONE classifier are addressed in Schmolze and Lipkis (1983). Krypton (Brachman et al., 1983) opts for a simpler language in order to guarantee completeness for the subsumption algorithm.



"A MESSAGE with exactly one Recipient, and all of whose Senders are STARFLEET-COMMANDERs."

Figure 7. Before classifying the Concept X.

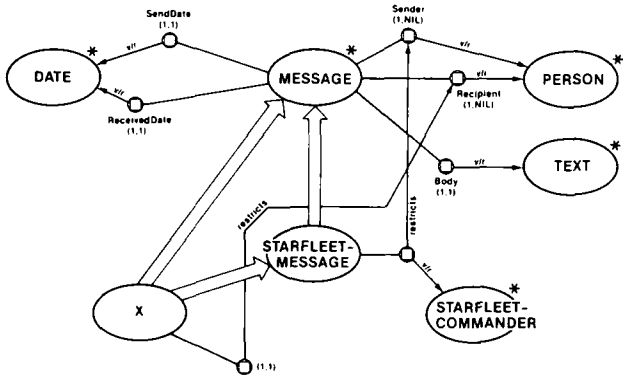


Figure 8. After classifying Concept X.

edge bases (because the system takes some of the work out of the user's hands), it supports dynamic creation of descriptions (Concepts) during the execution of some task.

In addition, the classifier performs a class of inferences that has been found to be very useful for several AI applications. A typical use of KL-ONE is the following. First, a static knowledge base is created that contains general information. Then, a reasoning task is begun that creates many new descriptions as representations of partial results. These descriptions are classified, and the discovered subsumers and/or subsumees are used to continue the reasoning process. The Appendix treats an example of this in some depth.

An important use of the classifier is for generalized search. If one forms a search pattern into a Concept (call it *PATTERN*), classification will

discover other Concepts that *PATTERN* subsumes. If the target of the search is also described by some Concept (call it *TARGET*), and if the pattern matches the target, then *PATTERN* will subsume *TARGET*. Hence, the first phase of a search process can be accomplished by using classification to restrict the search-space of possible target descriptions.

The classifier is an important contribution of KL-ONE, and we will demonstrate its utility with an extended example in the Appendix. The classifier's algorithm will not be described in this paper (but see Lipkis & Mark, 1981; Schmolze & Israel, 1983; Schmolze & Lipkis, 1983).

7. A NOTE ON DEFAULT VALUES AND CANCELLATION

As has been stated throughout this paper, all components of a Concept specify (at least) necessary conditions for individuals that the Concept can describe. This also applies to components that a Concept inherits from its subsumers. Thus, it would be inappropriate to "cancel" an inherited component and KL-ONE does not allow any such cancellation.

The lack of cancellation of Value Restrictions might appear problematic from the point of view of representing "exceptions" (e.g., three-legged elephants—see Fahlman, 1979). However, if we were to allow cancellation of components within Concepts, then these components would be reduced in status from necessary conditions to default assertions. We feel that such nonnecessary conditions are more appropriately expressed outside of the taxonomy. Furthermore, cancellation would derail the classifier. For example, the classifier would have its hands tied if Roles expressed defaults: A given Concept could be forced to fit almost anywhere, because all we would need to do is cancel the Roles that don't match up. In this way, *THREE-LEGGED-ELEPHANT* could just as well subsume *FOUR-LEGGED-ELEPHANT* as be subsumed by it. See Brachman (in press, a) for more about this problem.

We intend, instead, to allow statements of default rules *between Concepts* only. Thus (when implemented), one would *not* represent elephants' typically having four legs as in Figure 9. Instead, one would assert something like

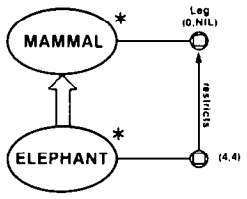


Figure 9. Not the way to describe elephants.

$$\frac{\text{Elephant(x) : M[Four-legged-mammal(x)]}}{\text{Four-legged-mammal(x)}}$$

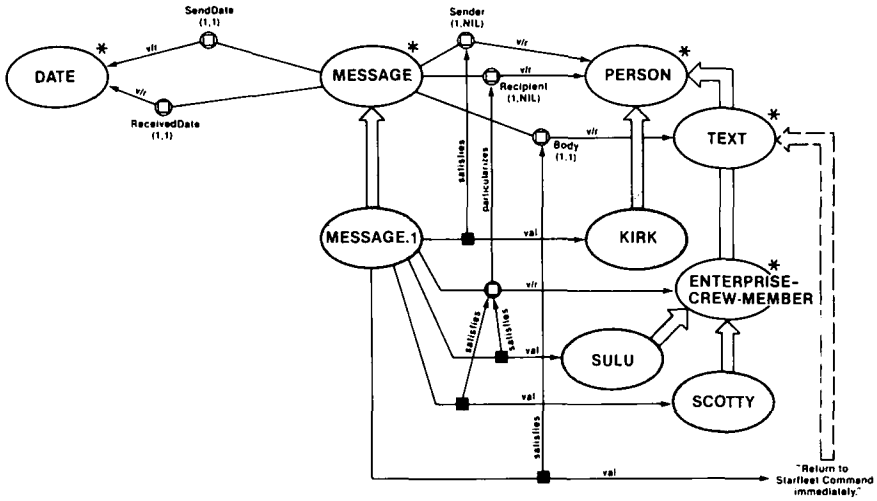
in the manner of Reiter (1980). That is, "unless you have information to the contrary, assume of an elephant that it is also a four-legged-mammal." This leaves the Concepts of *ELEPHANT* and *FOUR-LEGGED-MAMMAL* distinct (as they should be) and inviolate. KL-ONE seems to be different from many of today's representation languages precisely because of its reliance upon necessary conditions rather than default assertions.

8. INDIVIDUAL CONCEPTS

KL-ONE provides structures that are suitable for uniquely describing individuals. The primary unit for individual description is the *Individual Concept,* which is similar to the Generic Concept but can be used to describe at most one individual in a particular context. As with Generic Concepts, nothing is asserted about any particular individual when an Individual Concept is formed. Actual use of Concepts to make statements is the responsibility of the assertion language (section 10).

Each Individual Concept must individuate some Generic Concept, as does *MESSAGE.1* in Figure 10 (shaded ellipses in the Figure denote Individual Concepts, shaded arrows denote the *Individuates* link). The Individual Concept carries the same meaning as the Generic Concept it individuates, together with the fact that there can be at most one individual described by it per context (i.e., if there are two individuals, x and y, such that an Individual Concept describes both x and y; then $x = y$).

An Individual Concept also has associated Role descriptions that serve to describe the actual individual fillers of the Roles inherited from its parent Concept. There are two kinds of Roles that can be components of Individual Concepts: *IRoles* and *Particular RoleSets.* An IRole represents the binding of two individuals together in a relation. The relation is the one denoted by the parent RoleSet of the IRole (IRoles are always descended from Generic RoleSets). The two individuals are the one represented by the Individual Concept and the one that is described as filling the IRole. IRoles are the way to instantiate with arguments the two-place relations represented by RoleSets. For example, in Figure 10, there is an IRole (depicted as a filled-in square) that corresponds to the *Sender* RoleSet of *MESSAGE.* The link connecting the two is called the "satisfies" link. The other labeled link emanating from the IRole is called "val," and it leads to a description of the sender of *MESSAGE.1.* (The unlabeled link connecting the Individual Concept to the IRole is the "has Role" link.) Thus, if some individual is



"The MESSAGE with a Sender that is KIRK, a Body that is "Return to Starfleet Command immediately.", whose Recipients are all ENTERPRISE-CREW-MEMBERs, one of which is SULU, and one of which is SCOTTY."

Figure 10. An Individual Concept.

described by *MESSAGE.1,* its *Sender* will be described by the Individual Concept *KIRK. KIRK* is simply an individual *PERSON* about which (for simplicity) we offer no further information.

A Particular RoleSet is to an Individual Concept just as a Generic RoleSet that restricts a parent RoleSet is to a Generic Concept. It represents the set of fillers of the Role for the particular individual rather than some generic set of fillers. It has associated further restrictions upon fillers of the relation it represents, just as the Generic RoleSet does. These restrictions constrain all fillers of the Role and are taken conjunctively with restrictions inherited from the parent Role. For example, in Figure 10, a Particular RoleSet of *MESSAGE.1* further restricts the *Recipient* RoleSet of *MESSAGE* by adding a new Value Restriction—*ENTERPRISE-CREW-MEMBER*. For this *MESSAGE.1,* there are at least two recipients, one of which is described by *SCOTTY*, and one by *SULU.* Note that the IRoles, specifying these particular bindings, are descended from the Particular RoleSet and that there is exactly one for each filler (IRoles intrinsically have cardinality 1, whereas Particular RoleSets are like other RoleSets and have Number Restrictions). There is no restriction against more recipients for *MESSAGE.1* because the Number Restriction is (1,NIL). If the Particular RoleSet had constrained the Number Restriction to be (2,2), then all recipients would be accounted for.

It should be noted that Individual Concepts, as described here, are primitive. Their Role filler descriptions specify necessary conditions, but there are

no sufficient conditions for uniquely determining the referent of an Individual Concept.

As a final note, we should mention that in the KL-ONE implementation a Role filler can also be a Lisp object. The IRole for *Body* shows a Lisp string as the description of the *Body* of *MESSAGE.1*. In our KL-ONE implementation, each Lisp object is treated as an Individual Concept of a *Lisp-Type* Concept. In this case, there is an implicit "individuates" link from the string to the Concept TEXT.

9. STRUCTURAL DESCRIPTIONS

In earlier sections we covered some of the ways that Generic Concepts can be formed from their superConcepts by adding restrictions. Each of the types of "essential difference" presented so far (Value Restriction, Number Restriction, etc.) has involved restricting only a single Role at a time. A moment's thought about descriptions that occur in realistic knowledge bases reveals that we need a facility to form Concepts by constraining the relation between more than one Role of the same Concept.

The compositional apparatus introduced so far yields Concepts that, for all intents and purposes, have all of their Roles independent of one another. But, generally speaking, the functional roles that we want to represent are *inter*dependent. For example, the vertical clearance of an arch is a function of the location of its lintel and the surface the arch is standing on; or, we might characterize an "important-message" as one whose sender is the supervisor of the recipient. In KL-ONE, these kinds of relations among Roles are specified with *Structural Descriptions*.

The need to handle the various possible relations among Roles makes the technical details of Structural Descriptions (SDs) a bit messy. However, the intent is straightforward—an SD allows the formation of a description whose essential difference with its proximate genus is a relationshp among more than one of its Roles.

Before we present some of the details of the two types of SDs currently in KL-ONE, we should mention another motivation for their existence. While KL-ONE Roles can be given "names," these are meaningless strings as far as the system is concerned. In the structure presented before this section, we have seen how Roles describe actual or potential fillers, but nothing (except our wishful thinking) gives a Role its intended meaning as the description of a functional role to be played. In addition to providing a way to specify a new Concept whose difference from its parent is a constraint between Roles, SDs can add substance to the names attached to Roles. For example, the buyer in a transaction is the person to whom goods go in exchange for legal tender provided by that buyer. The Structural Description mech-

anism allows us to describe such a transaction in terms of two connected giving events (the giving of money and the giving of goods in exchange); the relation of the *Buyer* Role to the *Giver* and *Receiver* Roles of those giving events defines the role being played by the buyer in the transaction.[22]

9.1 Role Value Maps

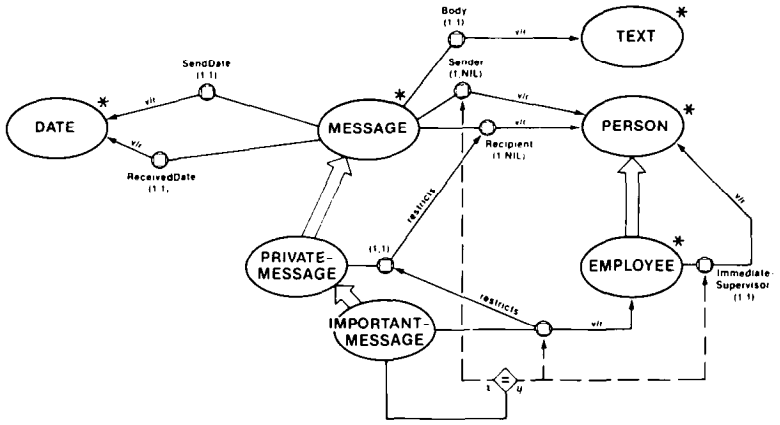
As mentioned, there are actually two types of SDs in KL-ONE. The simpler of these is called the *Role Value Map (RVM).* This special kind of Structural Description was introduced into the system for convenience. In the course of the use of KL-ONE in a natural language understanding system, it was often necessary to express the equality of two sets of Role fillers. Such relations could presumably be expressed with the more general SD facility, but would tend to involve more complex notation. So KL-ONE introduced a special notation to allow one to say things like "the grandparents of a person are exactly the same as the parents of the parents of that person."[23]

The crucial representational ingredient needed here is something that allows access to the Roles of a Concept from within an RVM. That is, to express equality between two Roles' fillers, we need some notation that allows us to access those Roles. Thus, the heart of an RVM resides in two pointers to Roles, or *Role Chains,* that are taken to stand for the sets of fillers of the Roles in an instance.

Figure 11 illustrates the simple structure of a Role Value Map. It shows the *PRIVATE-MESSAGE* and *IMPORTANT-MESSAGE* Concepts; the latter is intended to represent messages from the immediate supervisors of their recipients. The RVM is drawn as a diamond, and its two pointers, x and y, access the Roles whose fillers are to be equated (the Role Chains are drawn as dashed lines). In this case, the x pointer stands for the set of the senders of an instance of *IMPORTANT-MESSAGE* (in this case there will always be only one filler of the *Sender* Role), and the y pointer indicates the immediate supervisors of the recipients of the message (again, we expect only one). This latter Role Chain indicates how Roles can be composed to form constraints on embedded Role descriptions. One can think of Role Chains as a varia-

[22] In light of this, it is easy to see that the Role Differentiation mechanism of KL-ONE discussed in section 5.2 allows only the *primitive* derivation of new Roles from old ones. That is, if I specify the role of *BUYER* of a *TRANSACTION* as a subRole of *Participant* of an *ACTIVITY* by differentiation, I know nothing about what makes the filler the buyer (rather than, say, the seller). We have always envisioned a definitional mechanism whereby we could specify completely what it means to be a buyer (in terms of the other participants and goods in the transaction). SDs are the beginning of a mechanism sufficient to do this, but there is still a long way to go before Roles can be fully defined in KL-ONE.

[23] We have also implemented special-purpose routines for processing this set of relations.



"An IMPORTANT-MESSAGE is a PRIVATE-MESSAGE whose Recipient is an EMPLOYEE, and whose Sender is the same as the ImmediateSupervisor of its Recipient."

Figure 11. A Role Value Map.

tion of functional composition where the functions are set-valued, e.g., "ImmediateSupervisor(Recipient (IMPORTANT-MESSAGE))".

Note that the Role Value Map is strictly a part of *IMPORTANT-MESSAGE*, even though one of the Roles it accesses comes from *MESSAGE*. The Role Value Map is the essential difference between *IMPORTANT-MESSAGE* and *PRIVATE-MESSAGE*. Because subConcepts always inherit the Roles of their superConcepts, the constraint can use the *Sender* and *Recipient* Roles at *IMPORTANT-MESSAGE*, but it does not affect them, except in the context of *IMPORTANT-MESSAGE*.[24]

A Role Value Map specifies a necessary and sufficient condition. Thus, the RVM in *IMPORTANT-MESSAGE's* specification requires that each instance of IMPORTANT-MESSAGE satisfy the following: The set of persons that are the senders of the particular *MESSAGE* is the same set as the immediate supervisors of the recipients of that very same *MESSAGE*. Any *PRIVATE-MESSAGE* that satisfies this constraint is, by definition, an *IMPORTANT-MESSAGE*. The converse is also true—any *IMPORTANT-MESSAGE* satisfies the constraint.

KL-ONE allows a variation on the kind of RVM illustrated in Figure 11 that specifies a *subset* relation between sets of Role Chain fillers. It is depicted by a diamond surrounding a set inclusion symbol, and in such a case the set indicated by the x pointer is taken to be a subset of the one indicated by the y pointer.

[24] This is a place where the particular graphical notation may be more of a hindrance than a help.

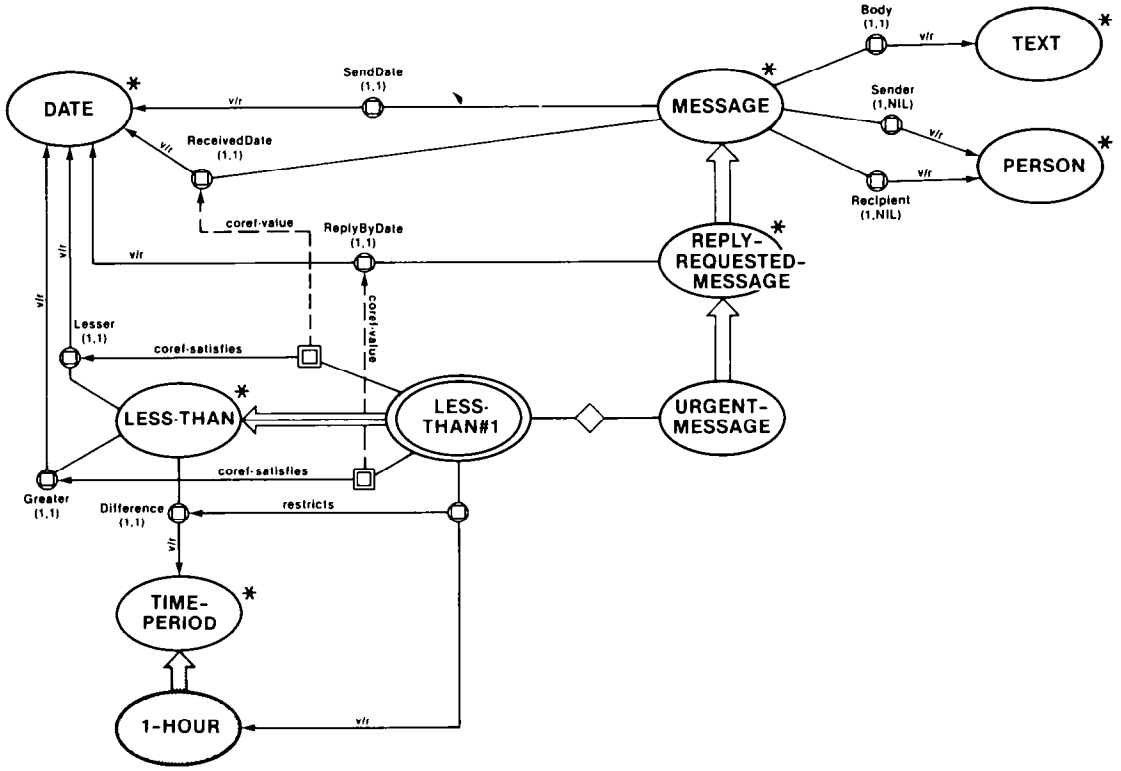
9.2. Structural Descriptions

The second, more general type of Structural Description allows us to express how the Roles of a Concept interrelate (and how they relate to the Concept as a whole) in terms of other Concepts in the network. Rather than just express a subset or equality relation between sets of Role fillers, these SDs can relate Roles in arbitrary ways by using Concepts defined elsewhere.

There are two fundamental aspects to the relation of two Roles using KL-ONE Concepts. First, there is simply getting access to those Concepts in such a way that their use in defining a new Concept does not inadvertently change their meaning or assert the existence of any individuals. One can make an analogy here to a similar phenomenon in programming languages: Functions can be defined in one place and used in definitions of other functions. This analogy points up the second aspect of KL-ONE's SD mechanism: Once we have an embedded "call" to a Concept, we need to bind the formal arguments of the called Concept to the actual arguments to be used in the context of the call. While programming languages typically use argument order to achieve the correspondence between actuals and formals, KL-ONE's philosophy advocates using explicit links; thus the KL-ONE structure that implements SDs is unhappily complicated.

For an illustration of some of the details of SDs, consider Figure 12, wherein we define the concept of an "urgent message" as one that requires response within one hour. We do this in terms of our familiar *MESSAGE* Concept and a Concept called *"LESS-THAN,"* which we presume is accounted for elsewhere in the network. The idea is to use the *LESS-THAN* to express a relation between the *ReceivedDate* of an *URGENT-MESSAGE,* and a new Role that we will call *"ReplyByDate."* The received-date will have to be less than 1 hour before the reply-by-date. For modularity, we introduce the new *ReplyByDate* Role at a Concept called *"REPLY-REQUESTED-MESSAGE"* ("among other things" appears in the specification of this Concept, because we are presuming that there is more to requesting a reply than adding a single field to a message).

The way that the KL-ONE structure in the figure expresses the relation we need is the following: the SD (the diamond in the figure) has associated with it a version of the *LESS-THAN* Concept. The structure of this internal version of *LESS-THAN* is isomorphic to that of the regular, Generic version. Because, however, its use is to be restricted to this particular definition of *URGENT-MESSAGE,* it itself is not a Generic but rather a version of the Concept "parameterized" by the surrounding context (the rest of the *URGENT-MESSAGE* structure). It is this *Parametric Individual Concept* (*LESS-THAN#1*) that represents the "call" to *LESS-THAN* within *URGENT-MESSAGE.*



"A REPLY-REQUESTED-MESSAGE is, among other things, a MESSAGE with a ReplyByDate, which is a DATE."

"An URGENT-MESSAGE is a REPLY-REQUESTED-MESSAGE whose ReceivedDate and ReplyByDate satisfy a LESS-THAN whose Lesser is the ReceivedDate, whose Greater is the ReplyByDate, and whose Difference is 1-HOUR."

Figure 12. A Structural Description that uses another Concept.

Once we have the internal version of *LESS-THAN* to work with, all we need is to bind the "actual" Roles to the "formal" ones. This is achieved by means of Role Chains, exactly as we saw with Role Value Maps in the previous section. In this case, we bind the *ReceivedDate* to the *LesserRole* of *LESS-THAN#1,* and the *ReplyByDate* to its *Greater* Role. *LESS-THAN#1,* its Roles, and some of the links in the SD are drawn slightly differently than they are in the case of *LESS-THAN,* because their function is somewhat different than in the Generic Case. *LESS-THAN#1* is drawn as a double ellipse, and its relation to *LESS-THAN* is shown by a wide, 3-line arrow that depicts the *parametric individuates* relation. The Roles of *LESS-THAN#1* are defined by *Coref-Satisfies* links to the corresponding RoleSets of *LESS-THAN,* and their bindings are defined by *Coref-Value* Role Chains.

We should add one more technical note. In Figure 12, the Coref-Value links point directly from Roles of *LESS-THAN#1* to RoleSets of *URGENT-MESSAGE* (actually, they point to RoleSets that *URGENT-MESSAGE* inherits). In general, Coref-Value links can be Role Chains and have the same properties as Role Chains for RVMs. When used with Parametric Individual Concepts, the Role Chain can also point directly to the enclosing Concept in order to express the participation of the instance's "self"—that is, the thing as a whole—in a relationship.

10. ASSERTIONAL LANGUAGE

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

One asserts the existence of some thing satisfying a description (i.e., Concept) by connecting it to a *Nexus* within a particular *Context.* This connecting link is called a *Description Wire.* A Nexus is a structureless entity which serves as a locus of coreference statements; it holds together various descriptions, all of which are taken to specify the same object in the Context. Nexuses have been conveniently thought of as corresponding to things in the world; KL-ONE, however, makes no such commitment. The Description Wires are also taken to be in the Context. Contexts are at the moment simply collections of Nexuses and Description Wires. Thus, a Context can

act as a "world," which comprises a set of statements about existence and description coreference.[25]

In Figure 13 (the Nexuses are small circles, the Contexts rectangles, and the Description Wires squiggly lines), we have Nexus N1 in Context C1 asserting that a Vulcan named Spock is the First Officer of the Enterprise, whereas in Context C2 these same descriptions are used in a different way by Nexuses N2 and N3 to assert that the First Officer of the Enterprise is a person named Uhura and a Vulcan named Spock is the Captain of the Enterprise. We should note that KL-ONE at the moment does not support any meaningful relations between Contexts, although a hierarchy of Contexts can be created by putting the meta-anchor (i.e., a Nexus—see section 11.1) of one Context into another Context.

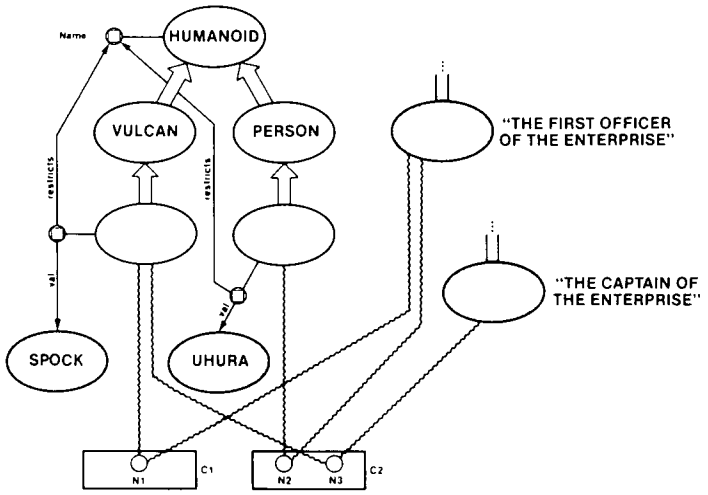


Figure 13. Some KL-ONE assertions.

11. ADDITIONAL KL-ONE FACILITIES

11.1 Metadescription

Nexuses allow us to come as close to reference to objects outside the system as is possible in this kind of representation environment. In addition to the use of Nexuses as surrogates for outside entities, KL-ONE allows reference to internal entities (e.g., Concepts) as well. Thus one can *metadescribe* a KL-ONE object *in KL-ONE*. Of course, to do this, the system needs to have the Concepts of a KL-ONE Concept, a KL-ONE Role, a KL-ONE Role Value Map, etc. These are not yet part of the implemented system.

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

In order to construct a metadescription, one uses the same type of structure used in constructing a regular description. Each KL-ONE structure is considered implicitly to have a corresponding Nexus that is known to exist in a "KL-ONE base level" Context.[26] Metadescriptions are simply descriptions (usually expressed in terms of the Concepts *KL-ONE-CONCEPT, KL-ONE-ROLE,* etc.) attached to those Nexuses by means of the Description Wire mechanism mentioned earlier. In the future, we expect to study how further to exploit metadescription in KL-ONE. The KL-ONE system might provide automatic access to a complete metadescription of any other description and also allow one to affect the KL-ONE interpreter by restricting these metadescriptions in a manner similar to that of Brian Smith's (1982) 3-LISP. The primary questions in this effort deal with the details of such a system, and more importantly, the determination of exactly what leverage one gains by using it.

11.2 Attached Procedures and Data

The final feature of KL-ONE to be touched on here is the ability to attach procedures and data to structures in the network. This is purely a programming convenience—attached procedures and data are outside of KL-ONE and have no semantically justifiable place in the epistemology. Hence, this section deals strictly with our implementation.

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

Procedures attached to the conceptual taxonomy can make KL-ONE work like a special kind of object-oriented programming system. We make no claims about this use of the system (but see Goodwin, 1979)—the procedures are not themselves written in KL-ONE, and there can be no guarantee that an attached procedure will honor the integrity of the network. The facility itself is supported only in a very simple way.

Finally, a facility has been incorporated to attach arbitrary data to KL-ONE Concepts. The data is stored in property list format and is inherited along superC cables. A second attached data facility exists which simply provides a property list format without inheritance.

[26] We have on occasion called these Nexuses *meta-anchors* in the manner of Smith (1978).

12. CONCLUSION

Work on KL-ONE continues, but much has been accomplished in the several years since its birth. Most importantly, the language has provided the basis for much further research and development in Artificial Intelligence and has helped focus a large number of people on some important facets of knowledge representation. It has also provided a practical foundation for a number of application systems.

On the technical side, KL-ONE has pioneered the idea of constructing a representation out of "epistemological" primitives and has provided a first set of such primitives for examination and experimentation. It has also instigated first-class status for Roles (a k a slots) in frame-based knowledge representation systems, including the potential for multiple fillers and explicit differentiation into subRoles. Further, KL-ONE has helped begin serious investigation of the separation of the representation task into descriptional and assertional components. It has also initiated serious research into the interaction of Roles through its Structural Description and Role Value Map mechanisms.

There is much more research to be done on representations derived from KL-ONE (i.e., Krypton and KL-TWO), but through it all, the kernel of KL-ONE survives.

REFERENCES

Amarel, S. (1968). On representation of problems of reasoning about actions. In D. Michie (Ed.), *Machine Intelligence 3*. Edinburgh: Edinburgh University Press.

Bobrow, R. J. (1979a). The RUS natural language parsing framework. In *Research in natural language understanding, annual report* (Report No. 4274). Cambridge, MA: Bolt Beranek and Newman.

Bobrow, R. J. (1979b). Semantic interpretation in PSI-KLONE. In *Research in natural language understanding, annual report* (Report No. 4274). Cambridge, MA: Bolt Beranek and Newman.

Brachman, R. J. (1977). What's in a Concept: Structural foundations for semantic networks. *International Journal of Man-Machine Studies, 9*, 127–152.

Brachman, R. J. (1978). *A structural paradigm for representing knowledge.* (Tech. Rep. No. 3605). Cambridge, MA: Bolt Beranek and Newman.

Brachman, R. J. (1979). On the epistemological status of semantic networks. In N. V. Findler (Ed.), *Associative networks: Representation and use of knowledge by computers.* New York: Academic.

Brachman, R. J. (1983). What is-a is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer, 16*(10), 30–36.

Brachman, R. J. (in press, a). I lied about the trees. *AI Magazine 6*(3).

Brachman, R. J. (in press, b). *A structural paradigm for representing knowledge.* Norwood, NJ: Ablex.

Brachman, R. J., Bobrow, R. J., Cohen, P. R., Klovstad, J. W., Webber, B. L., & Woods, W. A. (1979). *Research in natural language understanding, annual report* (Tech. Rep. No. 4274). Cambridge, MA: Bolt Beranek and Newman.

Brachman, R. J., Fikes, R. E., & Levesque, H. J. (1983). Krypton: A functional approach to knowledge representation. *IEEE Computer, 16*(10), 67-73.

Cohen, B. C. (1982). *Understanding natural kinds.* Unpublished doctoral dissertation, Stanford University, CA.

Donnellan, K. (1966). Reference and definite descriptions. *Philosophical Review, 75,* 281-304.

Fahlman, S. E. (1979). *NETL: A system for representing and using real-world knowledge.* Cambridge, MA: M.I.T. Press.

Fikes, R. E. (1982). Highlights from Klone Talk. In J. G. Schmolze & R. J. Brachman (Eds.), *Proceedings of the 1982 KL-ONE Workshop, BBN Report No. 4842.* Cambridge, MA: Bolt Beranek and Newman. (Also published as FLAIR Tech. Rep. No. 4, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA).

Freeman, M. W., & Tomlinson, C. J. (1982). Towards a calculus of structural descriptions. In J. G. Schmolze & R. J. Brachman (Eds.), *Proceedings of the 1981 KL-ONE Workshop, BBN Report No. 4842.* Cambridge, MA: Bolt Beranek and Newman. (Also published as FLAIR Tech. Rep. No. 4, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA).

Freeman, M., Hirschman, L., McKay, D., Miller, F., & Sidhu, D. (1983). Logic programming applied to knowledge-based systems, modeling and simulation. In *Proceedings of Conference on Artificial Intelligence.* Oakland University, Rochester, MI.

Freeman, M., Hirschman, L., McKay, D., & Palmer, M. (1983). KNET: A logic-based associative network framework for expert systems. (Tech. Rep.). Paoli, PA: Research and Development Division, SDC—A Burroughs Company.

Goodwin, J. W. (1979). *Taxonomic programming with Klone.* (Tech. Rep. LiTH-MAT-R-79-5). Linköping, Sweden: Informatics Laboratory, Linköping University.

Hendrix, G. G. (1979). Encoding knowledge in partitioned networks. In N. V. Findler (Ed.), *Associative networks: Representation and use of knowledge by computers.* New York: Academic.

Israel, D. J. (1983). On interpreting network formalisms. *Computers and mathematics with applications—special issue on computational linguistics, 9*(1), 1-14.

Israel, D. J. (1984). A short companion to the naive physics manifesto. In J. Hobbs & R. Moore (Ed.), *Formal theories of the common sense world.* Norwood, NJ: Ablex.

Israel, D. J., & Brachman, R. J. (1984). Some remarks on the semantics of representation languages. In M. L. Brodie, J. Mylopoulos, J. W. Schmidt (Eds.), *On conceptual modelling: Perspectives from artificial intelligence, databases, and programming languages.* New York: Springer Verlag.

Kripke, S. A. (1980). *Naming and necessity.* Cambridge, MA: Harvard University Press.

Lipkis, T., & Mark, W. (1981). *Consul note 5, The consul classifier.* Marina del Rey, CA: USC/Information Sciences Institute.

McDermott, D. (1982). Artificial intelligence meets natural stupidity. In J. Haugeland (Ed.), *Mind design.* Cambridge, MA: MIT Press.

Moser, M. G. (1983). An Overview of NIKL, The new implementation of KL-ONE. In C. Sidner, M. Bates, R. Bobrow, B. Goodman, A. Haas, R. Ingria, D. Israel, D. McAllester, M. Moser, J. Schmolze, M. Vilain, *Research in knowledge representation for natural language understanding, annual report* (BBN Report No. 5421). Cambridge, MA: Bolt Beranek and Newman.

Nilsson, N. J. (1980). *Principles of artificial intelligence.* Palo Alto, CA: Tioga.

Quine, W. V. O. (1960). *Word and Object.* Cambridge, MA: M.I.T. Press.

Reiter, R. A. (1980). A logic for default reasoning. *Artificial Intelligence, 13,* 81-132.

Rifkin, A. (1985). *A deontic model of natural categories: featural definitions, cross-classification, and context sensitivity.* Unpublished doctoral dissertation, City University of New York, NY.

Schmolze, J. G., & Brachman, R. J. (1982, June). *Proceedings of the 1981 KL-ONE workshop.* (Tech. Rep. No. 4842). Cambridge, MA: Bolt Beranek and Newman. (Also pub-

lished as FLAIR Tech. Rep. No. 4, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA.)

Schmolze, J., & Israel, D. (1983). KL-ONE: Semantics and classification. In C. Sidner, M. Bates, R. Bobrow, B. Goodman, A. Haas, R. Ingria, D. Israel, D. McAllester, M. Mosler, J. Schmolze, & M. Vilain, *Research in knowledge representation for natural language understanding, annual report, (BBN Report No. 5421)*. Cambridge, MA: Bolt Beranek and Newman.

Schmolze, J. G., & Lipkis, T. A. (1983). Classification in the KL-ONE knowledge representation system. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. Karlsruhe, W. Germany.

Schubert, L. K., Goebel, R. G., & Cercone, N. J. (1979). The structure and organization of a semantic net for comprehension and inference. In N. V. Findler (Ed.), *Associative networks: Representation and use of knowledge by computers*. New York: Academic.

Sellars, R. W. (1917). *The essentials of logic*. Cambridge, MA: The Riverside Press.

Sidner, C. L. (1979). *Towards a computational theory of definite anaphora comprehension in English discourse*. (Tech. Rep. AI-TR-537). Cambridge, MA: Artificial Intelligence Laboratory, M.I.T.

Sidner, C. L., Bates, M., Bobrow, R. J., Brachman, R. J., Cohen, P. R., Webber, B. L., & Woods, W. A. (1981). *Research in knowledge representation for natural language understanding, annual report*. (Tech. Rep. No. 4785). Cambridge, MA: Bolt Beranek and Newman.

Smith, B. C. (1978). *Levels, layers, and planes: The framework of a theory of knowledge representation semantics*. Unpublished master's thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Smith, B. C. (1982). *Reflection and semantics in a procedural language*. Unpublished doctoral dissertation, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Tranchell, L. M. (1982). *A SNePs implementation of KL-ONE*. (Tech. Rep. 198). Buffalo, NY: Dept. of Computer Science, State Univ. of New York at Buffalo.

Woods, W. A. (1975). What's in a link: Foundations for semantic networks. In D. G. Bobrow & A. Collins (Eds.), *Representation and understanding: Studies in cognitive science*. New York: Academic.

Woods, W. A. (1979). *Theoretical studies in natural language understanding, annual report* (Tech. Rep. No. 4332). Cambridge, MA: Bolt Beranek and Newman.

Woods, W. A. (1983). What's important about knowledge representation? *IEEE Computer*, 1983, *16*(10), 22-27.

Woolf, B. (1982). An intelligent tutor for beginning programmers. In J. G. Schmolze & R. J. Brachman (Eds.), *Proceedings of the 1981 KL-ONE Workshop, BBN Report No. 4842*. Cambridge, MA: Bolt Beranek and Newman. (Also published as FLAIR Tech. Rep. No. 4, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA).

Zdybel, F., Greenfeld, N. R., Yonke, M. D., & Gibbons, J. (1981). An information presentation system. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. Vancouver: International Joint Conferences on Artificial Intelligence.

APPENDIX
AN EXAMPLE USING KL-ONE AND THE CLASSIFIER

In this appendix we present an example of a system using KL-ONE. From a natural language understanding (NLU) context (Brachman et al., 1979), we have chosen to describe a part of the process that translates English sentences into representations of their meanings.

The example has been chosen because it shows a system

- creating KL-ONE structures during a complex process,
- making use of classification, and
- using structures and procedures that go "outside" of KL-ONE in order to capture that which KL-ONE cannot.

We include the latter because the use of KL-ONE typically entails more than what can be "said" in KL-ONE per se. Woods (1983) calls this type of use the "conceptual coat rack" approach, and we will see this as the example is developed.

The NLU system in question uses the RUS parser (Bobrow, 1979a) to translate English sentences into the equivalent of parse trees, and it uses PSI-KLONE (Bobrow, 1979b) to translate these parses into KL-ONE representations of the meanings of the corresponding sentences. PSI-KLONE proceeds in two phases. First, it translates parsed sentences into a more structured syntactic representation. From this, interpretation rules are used to create the actual meaning representations. The two representations used by PSI-KLONE are in KL-ONE.

Before proceeding, we note that we have simplified the description of PSI-KLONE for presentation purposes. Also, the output of PSI-KLONE is not a representation of the final meaning of the sentence but a *literal semantic interpretation* that becomes the input to the portion of the system that deals with pragmatics.

The KL-ONE representation of a sentence's syntactic structure is built by PSI-KLONE by making use of a *syntaxonomy*. This is a group of Concepts in a network, each denoting a class of sentence fragments. They are distinguished from each other by both grammatical considerations and the particular words used in the fragments. For example, *NP* is a Concept denoting noun phrases (based on grammar only) and *PERSON-NP* is a Concept denoting noun phrases that, in turn, denote people (based on both grammar and particular nouns that refer to people). With each Concept in the syntaxonomy, we associate one or more interpretation rules that map descriptions of sentence fragments into semantic representations. The semantic interpretation process for a sentence S is as follows:

1. Sentence S is parsed.
2. A representation of the parse vis-à-vis the syntaxonomy is created (call it S ′).
3. The KL-ONE classifier is invoked to find all legitimate superConcepts of S ′ that are not already known. In particular, we are interested in those superConcepts that are in the syntaxonomy because they have interpretation rules associated with them.
4. Some set of interpretation rules are now applicable to S ′, namely, those rules that are applicable to the superConcepts of S ′. By using inheritable attached data (see section 11.2) to store the interpretation rules, the KL-ONE system automatically calculates this set.
5. A special interpreter executes the interpretation rules to produce the semantic interpretation.

Thus, the Concepts in the syntaxonomy are used both as a discrimination net for determining which interpretation rules apply and as a mechanism for inheriting the appropriate interpretation rules. Furthermore, the classifier performs the bulk of the discriminating.

An interesting feature of RUS and PSI-KLONE is that all of these steps can proceed in parallel. When the parser has found a sentence fragment, it immediately passes it along to the process concerned with finding its analog in the syntaxonomy, while the parser returns to the remainder of the sentence. As soon as the fragment's analog in the syntaxonomy is found, its inherited interpretation rules are executed and the corresponding literal semantic representation is constructed. Thus, RUS and PSI-KLONE proceed simultaneously. The purpose of this parallelism is two-fold. First, it can be an effective use of low-level parallel hardware. Second, because the parse of a sentence is often semantically ambiguous and some parses may be semantically incoherent (such as "round square") we reduce the search space for semantically coherent interpretations with the following. Because the interpretation rules can detect incoherent sentence fragments, PSI-KLONE provides immediate feedback to the parser as soon as an incoherent fragment is detected. The parser, in turn, immediately dispenses with all possible parses that involve the incoherent fragment. If this were not done, the parser might generate many parses that all included the same incoherent fragment. So, by intermingling parsing and interpretation, the search space is reduced.

In the remainder of this section, we will demonstrate in some detail how KL-ONE helps significantly in the discrimination process, i.e., in finding a sentence's analog in the syntaxonomy. Space limitations will not allow us to show the use of interpretation rules. Our example sentence is:

"That professor teaches undergraduates about Lisp on Thursday."

Figure 14 shows a parse tree for it. The sentence is a clause with a logical subject that is a noun phrase (NP) consisting of a determiner "That" and noun "professor." The clause's head verb is the verb "teaches" and its logical object is the noun phrase "undergraduates." It also has two prepositional phrase (PP) modifiers, "about Lisp" and "on Thursday."

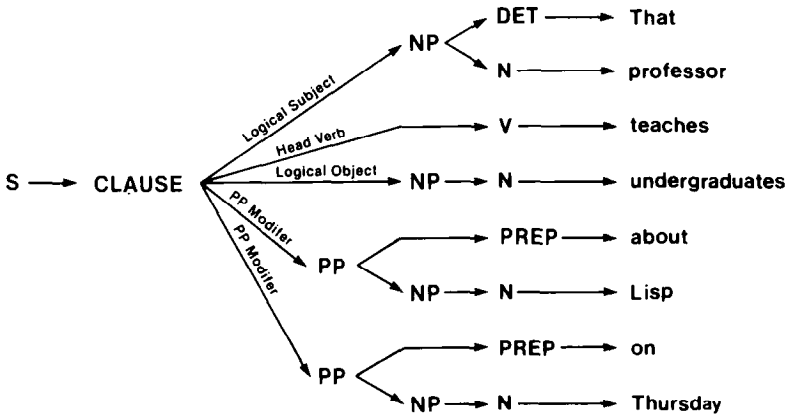


Figure 14. A parse tree for the example sentence.

In Figure 15, we depict a portion of the syntaxonomy that includes its most general Concepts. The Concept *PHRASE* is a primitive Concept that denotes all phrases. It has no further KL-ONE structure, so no more can be said about it. *PHRASE* has three immediate subConcepts, which are all primitive as well: *NP, CLAUSE,* and *PP.* These Concepts denote noun phrases, clauses, and prepositional phrases, respectively, all of which are types of phrases. We have drawn *NP* and *PP* with dashed lines to indicate that their entire KL-ONE structure is not shown here; we will show that later. Another primitive Concept is *VERB,* which denotes all verbs, and it

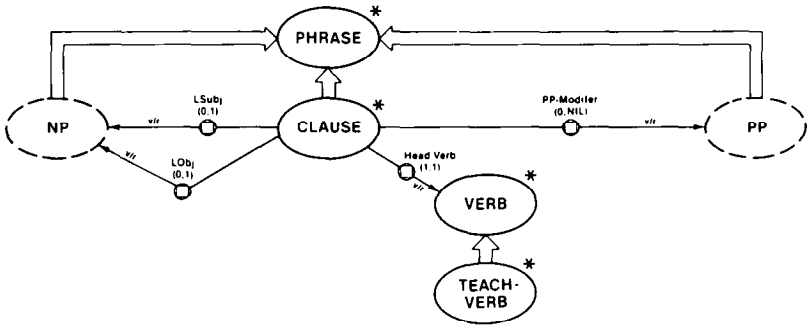


Figure 15. The top of the syntaxonomy: Phrases and Clauses.

has a primitive subConcept *TEACH-VERB,* which denotes all verbs that refer to teaching. Throughout this section, we will only include Concepts that pertain to our example sentence.

The Concept *CLAUSE* is shown in its entirety, and although it is primitive, it has several RoleSets. Thus, while these RoleSets specify necessary conditions for clauses, KL-ONE cannot express sufficient conditions for them. The RoleSet *LSubj* denotes the logical subject of a clause. It has a Value Restriction of *NP* and a Number Restriction of "(0,1)." This means that each *CLAUSE* need not have a logical subject, but if it does, it can have at most one. Also, each logical subject must be a noun phrase. The RoleSet *LObj* denotes the logical object of a clause. From its Value and Number Restrictions, we can see that each clause can have at most one logical object and each must be a noun phrase. *Head-Verb* denotes the head verb of a clause, and its Value Restriction is *VERB.* Thus, each clause must have exactly one head verb and each head verb must be a verb. The final RoleSet, *PP-Modifier,* denotes prepositional phrase modifiers. A clause can have any number of them, each of which must be a prepositional phrase.

The RUS parser begins by examining our example sentence from left to right while looking for certain sentence fragments and passing them along to PSI-KLONE. PSI-KLONE places a fragment's analog in the syntaxonomy and then builds the literal semantic interpretations of each fragment. The ability to build such interpretations is a test of semantic coherence (if the semantic interpreter fails, the fragment is incoherent); the result of the process is passed back to the RUS parser. RUS does not need to parse an entire sentence before calling upon PSI-KLONE, however, it does impose a certain order upon the fragments it sends:

1. It first parses enough of a sentence, which is a clause, to find a plausible head verb. PSI-KLONE is informed that a clause has been found with the given head verb and with the remaining constituents unspecified.
2. Next, RUS passes the logical subject of the clause to PSI-KLONE. If it must parse further in order to obtain the logical subject, it does so. Otherwise, it does so without further parsing. This strategy of parsing as needed is followed throughout.
3. The logical object is passed next.
4. Pre-modifiers of the clause are passed, from right-most to left-most.
5. Post-modifiers are passed, from left-most to right-most.
6. Finally, PSI-KLONE is informed that the clause is complete.

Getting back to our example, RUS first passes this message along to PSI-KLONE:

> A clause was found that will be named "cl#teaches." Its head verb is "teaches," which is a verb; cl#teaches is incomplete.[27]

PSI-KLONE will use 3 dictionaries. One dictionary maps terms in the communication language into terms in the syntaxonomy, such as mapping "clause" to the Concept *CLAUSE.* Another dictionary maps words to their morphological roots, such as mapping "teaches" to "teach." A third maps a particular word (that is a morphological root) along with its grammatical category into the syntaxonomy Concept that best describes it. For example, "professor" as a noun is best described by the Concept *TEACHER-NOUN.*[28]

Using these dictionaries, PSI-KLONE finds that clauses are represented by *CLAUSE* and it begins to construct an Individual Concept of *CLAUSE,* which it calls *CL#TEACHES.* PSI-KLONE also sees that "teaches" is a verb, and it looks up the morphological root of "teaches," which is "teach," and finds that it is best described by the Concept *TEACH-VERB.* So, it creates an Individual Concept of *TEACH-VERB* for "teaches," which it calls *VB#TEACHES.* This verb happens to be a complete fragment, so PSI-KLONE calls upon the classifier to find all legitimate superConcepts of *VB#TEACHES.*[29] However, because it has no structure other than its parent Concept, no new superConcepts are found. And as it happens, there are no interpretation rules associated with just verbs, so the verb is accepted as being semantically coherent, as there are no contraindications. Returning to the clause, PSI-KLONE finds the RoleSet *Head-Verb,* which indicates that the Value Restriction for head verbs of clauses is *VERB.* "Teaches" satisfies this restriction, so no problems are reported. It also checks the Number Restriction, which is satisfied. Now PSI-KLONE adds an IRole whose *val* is *VB#TEACHES,* and that *satisfies Head-Verb* from *CL#TEACHES.* However, because our clause is incomplete, PSI-KLONE does nothing more with *CL#TEACHES* and it returns to RUS with

> Clause cl#teaches is represented by the Individual Concept *CL# TEACHES.* It is OK so far. Its head verb is represented by the Individual Concept *VB#TEACHES,* which is complete and coherent.

[27] We use an English version of the actual communication language.

[28] This third dictionary contains information that may not seem suitable for a mechanism as simple as a dictionary. But keep in mind that the output of PSI-KLONE becomes the input to the part of the system that deals with pragmatics, and it is there that much of the complexity of language is dealt with. For example, if "professor" had actually been used to refer to a robot, then *TEACHER-NOUN* (which is constrained to apply to *people* who are teachers) would, in the end, be inappropriate. However, we allow this "mistake" for now and rely upon the pragmatics component to deal with it.

[29] Unfortunately, the actual implementation of the classifier does not work with Individual Concepts, so PSI-KLONE is forced to use only Generic Concepts for this task. However, extending the classifier to perform as explained in this section would be simple and straightforward.

RUS now knows the syntaxonomy analogs of the fragments it has passed, and RUS is responsible for keeping track of this. Figure 16 shows the current representation of *CL#TEACHES.*

Concepts for nouns and noun phrases are shown in Figure 17. On the top left side is the Concept *NOUN,* which denotes all nouns, and its descendant Concepts. *NOUN* is a primitive Concept without any further structure, and it has 3 immediate descendants, *TIME-NOUN, PERSON-NOUN,* and *SUBJECT-NOUN,* denoting nouns that describe time, people, or subjects, respectively (where subjects include history, computer science, Lisp, etc.). Each of these is primitive and without further structure. *PERSON-NOUN* is further distinguished by *STUDENT-NOUN,* denoting nouns that describe students, and *TEACHER-NOUN,* denoting nouns that describe teachers.

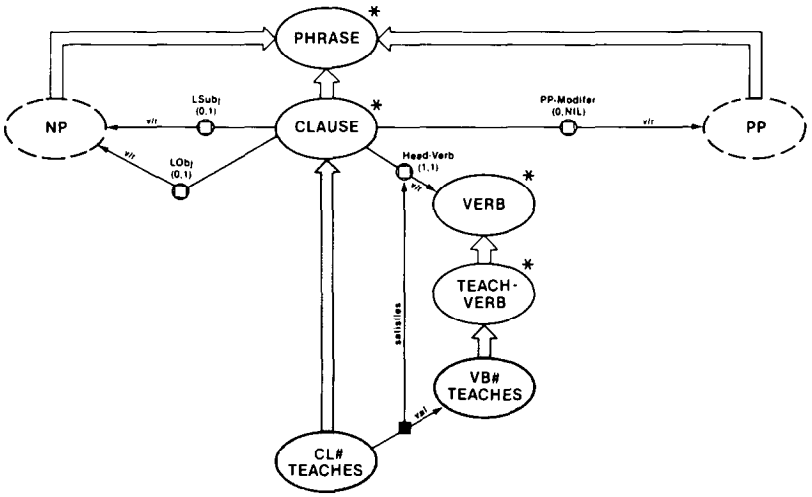


Figure 16. Partial representation of sentence: with head verb only.

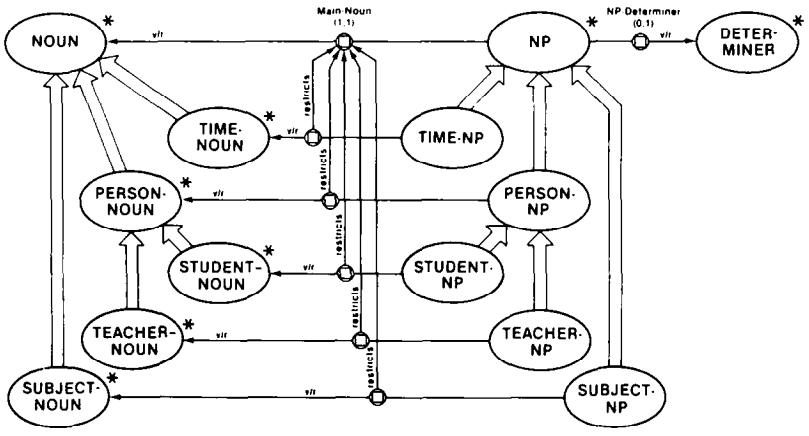


Figure 17. Part of the syntaxonomy: Nouns and Noun Phrases.

On the top right side of Figure 17 is the Concept *NP*, which denotes noun phrases. Although it is primitive, it has a RoleSet called *Main-Noun* whose Value Restriction is *NOUN* and Number Restriction is "(1,1)." *Main-Noun* denotes the main noun of a noun phrase, so each noun phrase must have exactly one main noun and it must be a noun. *NP* has another RoleSet *NP-Determiner*, which denotes the determiner of a noun phrase. The Number Restriction here states that there can be at most one determiner per noun phrase and it must be an instance of *DETERMINER*, a primitive Concept that denotes determiners.

A descendant of *NP* is *TIME-NP*, and this is our first defined Concept. Because it is defined, its complete meaning can be determined from the network. *TIME-NP* specializes *NP*, so each instance of *TIME-NP* must be a noun phrase. Furthermore, the RoleSet at *TIME-NP* restricts *Main-Noun* and has a Value Restriction of *TIME-NOUN*. Its Number Restriction of "(1,1)" is the same as that of its parent Concept, *NP*. Thus, each instance of *TIME-NP* is just a noun phrase whose main noun is a noun describing time. *PERSON-NP*, *STUDENT-NP*, *TEACHER-NP*, and *SUBJECT-NP* are specified similarly, and they are defined Concepts that denote noun phrases describing, respectively, people, students, teachers, and subjects.

Returning to our process, RUS passes along the logical subject of our sentence:

> Clause cl#teaches has a logical subject, to be named "np#prof," that is a noun phrase; np#prof has "professor" as its main noun and "That" as its determiner; np#prof is complete, but cl#teaches is not.

PSI-KLONE now constructs the Concept *DET#THAT* for "That," an Individual Concept of *DETERMINER*. For "professor," it constructs a Concept *N#PROF* that individuates *TEACHER-NOUN*, which it determines from its dictionaries. Finally, it constructs *NP#PROF*, an Individual Concept of *NP* with a *Main-Noun* of *N#PROF* and a *NP-Determiner* of *DET# THAT*. Since *DET#THAT*, *N#PROF*, and *NP#PROF* are complete, they are classified. For *DET#THAT* and *N#PROF*, no new information is discovered. However, the classifier finds that *NP#PROF* is also an Individual Concept of *TEACHER-NP*. Thus, it inherits interpretation rules that are applicable to such noun phrases, and we will assume that these rules generate a coherent interpretation. Finally, the Concept *CL#TEACHES* is expanded to include the logical subject, as shown in Figure 18. PSI-KLONE returns to RUS with:

> Clause cl#teaches is still OK. Its logical subject is represented by the Individual Concept *NP#PROF*, with a coherent interpretation of....

Here, the interpretation for the noun phrase "That professor" is passed back to RUS; we have not shown that interpretation.

In a similar fashion, RUS passes along the logical object "undergraduates" and PSI-KLONE creates its analog, *NP#UNDERGRADS*, and ex-

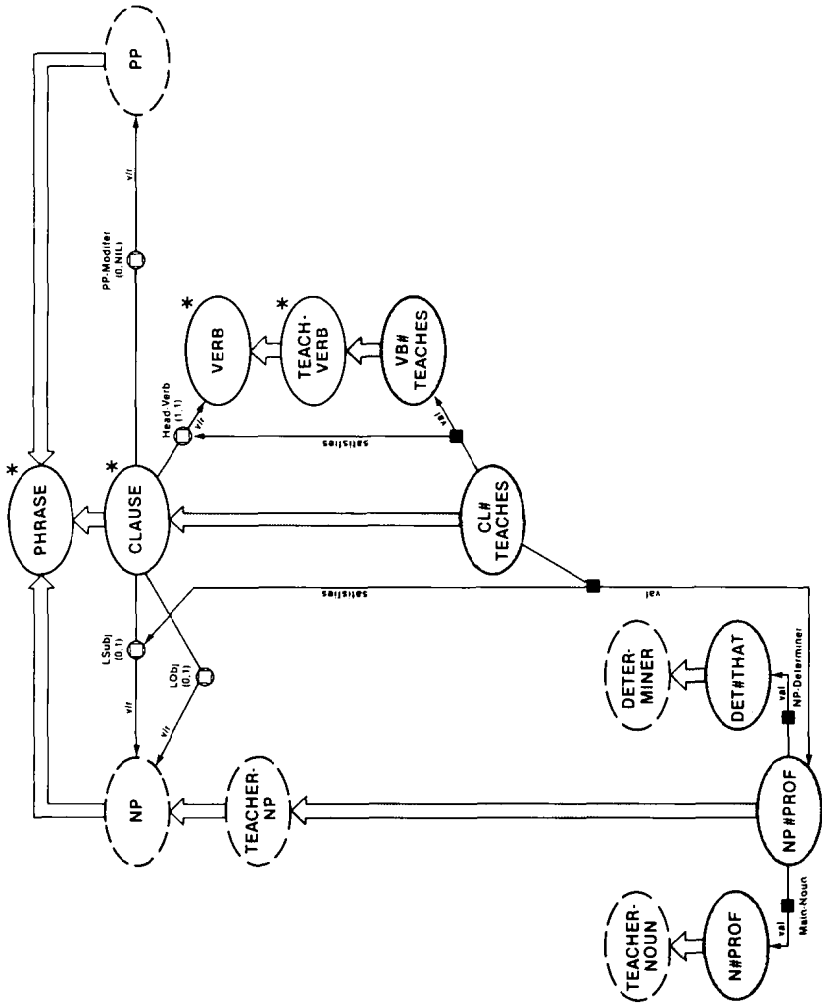


Figure 18. Partial representation of sentence: logical subject added.

pands the representation of *CL#TEACHES* as shown in Figure 19. Actually, the word "undergraduates" has an analog as a noun, *N#UNDERGRADS,* and as a noun phrase, *NP#UNDERGRADS.*

The syntaxonomy also includes Concepts for prepositional phrases, as shown in Figure 20. *PP* denotes all prepositional phrases (PPs), *PP-Prep* denotes the preposition of a PP, and *PP-Object* denotes its object. We can see that a PP must have exactly one preposition that must be a preposition, where *PREPOSITION* denotes all prepositions. Also, each PP must have exactly one object which must be a noun phrase. *ABOUT-PREPOSITION* denotes a singleton set, the preposition "about," and the various Concepts for types of noun phrases were described earlier.

Here there are several defined Concepts. *ABOUT-PP* denotes just those PPs whose preposition is the word "about." *TIME-PP* denotes just those PPs whose objects refer to time. *ABOUT-SUBJECT-PP* denotes just those PPs whose object refers to a subject and whose preposition is the word "about."

Now we can continue with the handling of the two PPs in our example. Using its dictionaries and the classifier, PSI-KLONE finds that "about Lisp" is an instance of *ABOUT-SUBJECT-PP,* which we will call *PP#LISP.* Similarly, "on Thursday" is found to be an instance of *TIME-PP,* which we will call *PP#THURS.* For the sake of brevity, in our discussion we have dispensed with the Individual Concepts for each of "about," "Lisp," "on," and "Thursday," and how they relate to *PP#LISP* and *PP#THURS.* In the actual system, of course, these are accounted for. This completes the clause, and simultaneously, the sentence, so RUS signals that the end of the clause has been reached. The representation of *CL#TEACHES* as it stands now is shown in Figure 21. However, before we show PSI-KLONE's final steps, we must first explain the final portion of the syntaxonomy, as shown in Figure 22.

Here we have specified several defined subConcepts of *CLAUSE* and we have used RoleSet differentiation for the first time in this section. *TIME-PP-CLAUSE* has a RoleSet called *Time-PP-Modifier* that differentiates the *PP-Modifier* RoleSet of *CLAUSE,* which means that *some* of a clause's PP modifiers can be also be instances of *Time-PP-Modifier.* The Value Restriction for *Time-PP-Modifier* is *TIME-PP,* which was shown earlier to denote PPs whose objects refer to time. Its Number Restriction is "(1,NIL)." Therefore, an instance of *TIME-PP-CLAUSE* is a clause that has at least one PP modifier that satisfies the constraints of *Time-PP-Modifier.* However, we must remember that RoleSet differentiation describes *necessary, but not sufficient* conditions. Thus, *KL-ONE* cannot independently recognize that two objects stand in the relation denoted by *Time-PP-Modifier,* just as it can't do the same for, say, *Head-Verb.* Only some outside sources can do

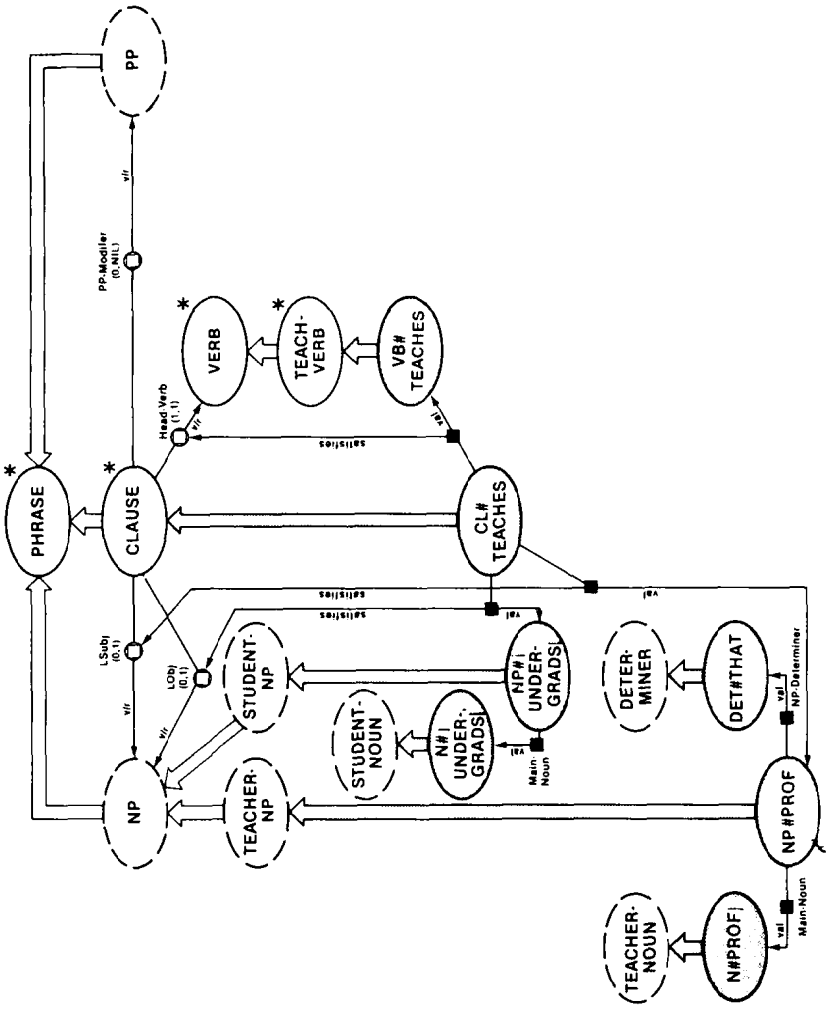


Figure 19. Partial representation of sentence: logical object added.

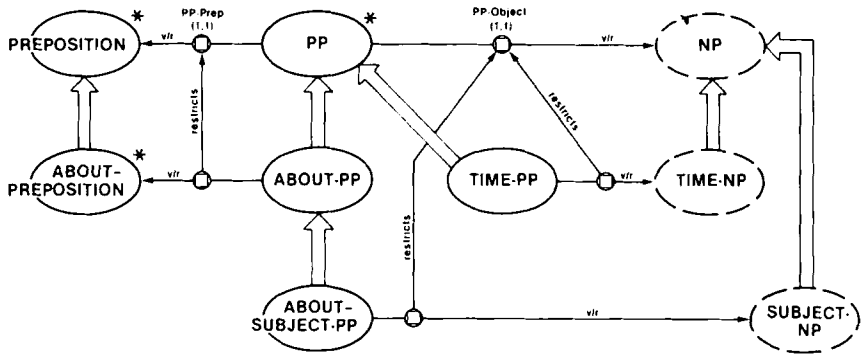


Figure 20. Part of the syntaxonomy: Prepositional Phrases.

that. However, our intended meaning of *Time-PP-Modifier* is exactly as if these conditions were sufficient, i.e., it denotes those clauses that have at least one PP modifier whose object refers to time. Unfortunately, we cannot say this presicely in KL-ONE.

ABOUT-SUBJECT-CLAUSE is specified similarly to *TIME-PP-CLAUSE*. The only difference here is that we differentiate *PP-Modifier* to form *About-Subject-PP-Modifier*. As with *Time-PP-Modifier*, our intended meaning is as if the necessary conditions represented in KL-ONE were sufficient as well, i.e., *ABOUT-SUBJECT-CLAUSE* denotes those clauses with at least one PP modifier whose preposition is "about" and whose object refers to subjects.

The meaning of *TEACH-STU-CLAUSE* follows easily. It denotes just those clauses with a head verb that refers to teaching and a logical object that refers to students.

So, before PSI-KLONE can classify the analog for the entire clause, *CL#TEACHES*, it must enforce our intended meanings for the RoleSets *Time-PP-Modifier* and *About-Subject-PP-Modifier*. In other words, whenever it determines that some PP is a modifier of a clause, it must also test whether it satisfies the relations denoted by those two RoleSets. This requires an additional mechanism in PSI-KLONE that we will not describe due to space limitations. However, given our intended meanings for these RoleSets, we can see that "about Lisp" satisfies the meaning of *About-Subject-PP-Modifier* and that "on Thursday" satisfies the meaning of *Time-PP-Modifier*.

We now classify *CL#TEACHES* and find it has several new parent Concepts, *TIME-PP-CLAUSE*, *ABOUT-SUBJECT-CLAUSE*, and *TEACH-STU-CLAUSE*. This is represented by forming a Concept that is just the conjunction of these parent Concepts, and having *CL#TEACHES* individuate

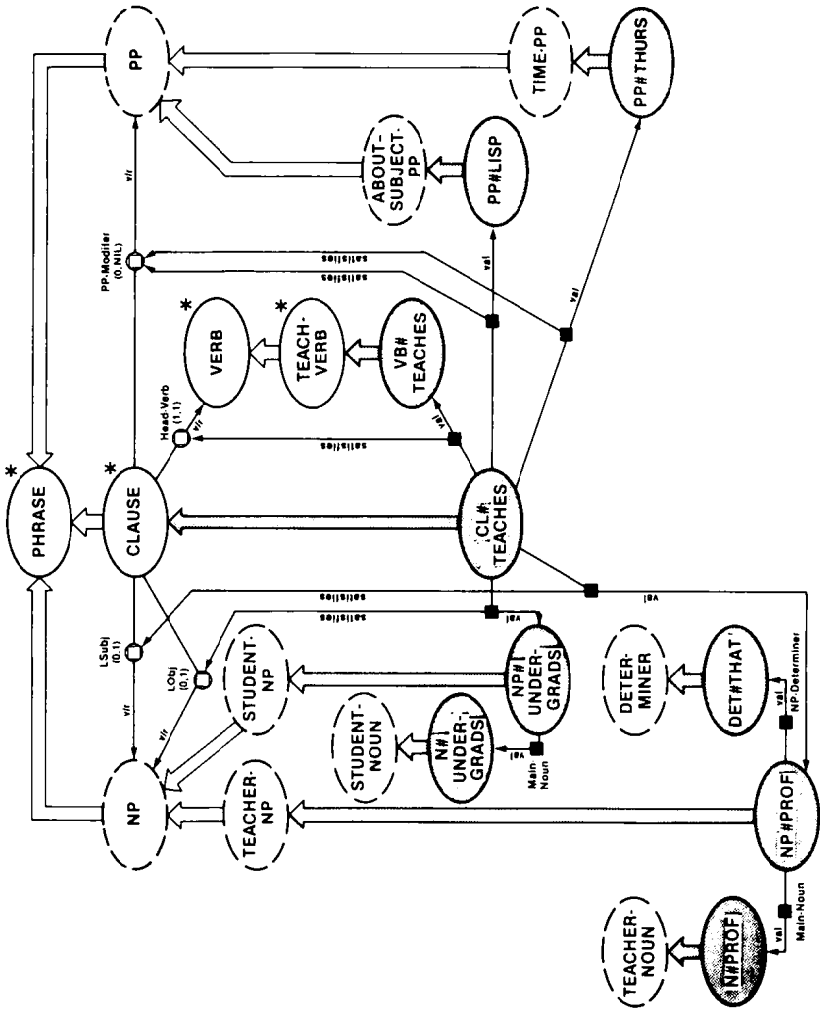


Figure 21. Representation of sentence with all constituents.

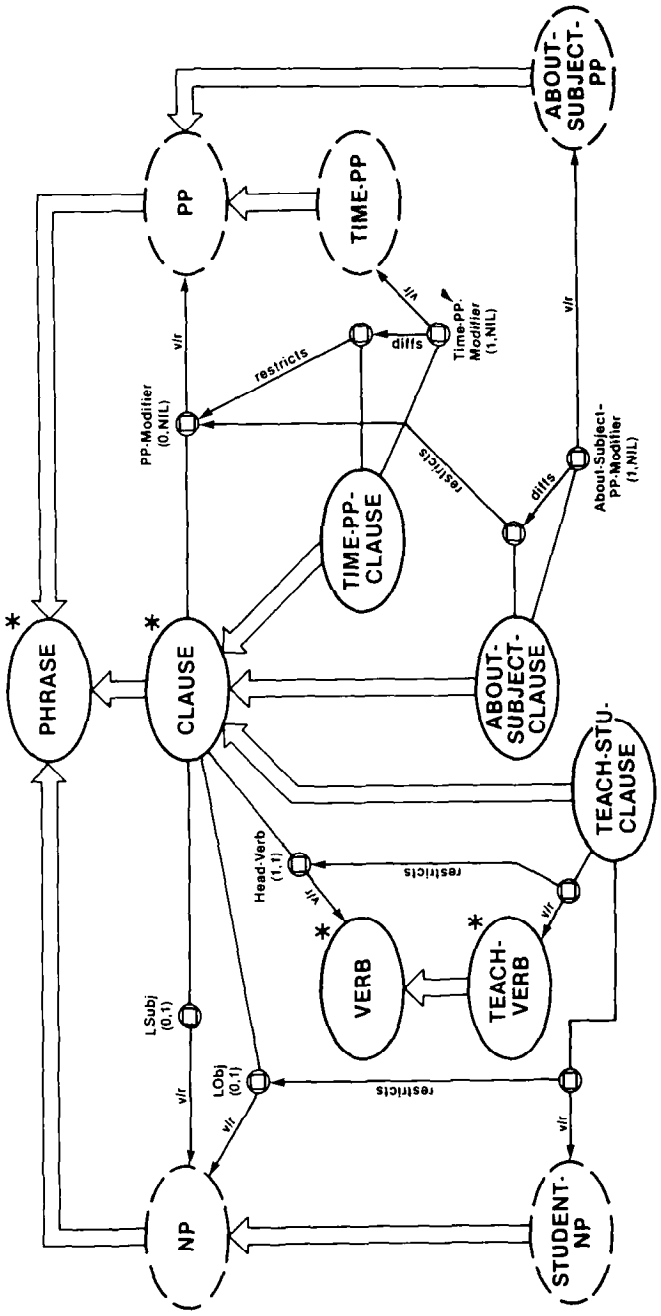


Figure 22. Part of the syntaxonomy: Clauses.

it. Figure 23 shows this; the unnamed Concept is defined to be just the conjunction of its parent Concepts.

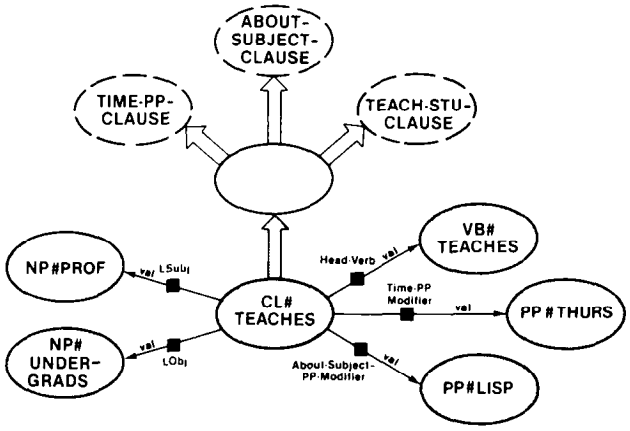


Figure 23. Final representation of sentence: after classification.

We assume that appropriate interpretation rules are inherited from the newly discovered parent Concepts and that PSI-KLONE makes a coherent interpretation. RUS and PSI-KLONE are now done with the sentence, and we are done with our discussion.