Knowledge Representation and Reasoning with Definitional Taxonomies

Robert A. Weida

Department of Computer Science
Columbia University

Technical Report CUCS-047-91

Abstract

We provide a detailed overview of knowledge representation issues in general and terminological knowledge representation in particular. Terminological knowledge representation, which originated with KL-ONE, is an object-centered approach in the tradition of semantic networks and frames. Terminological systems share three distinguishing characteristics: (1) They are intended to support the definition of conceptual terms comprising a "terminology" and to facilitate reasoning about such terms. As such, they are explicitly distinguished from assertional systems which make statements of fact based on some terminology. (2) Their concepts are arranged in a taxonomy so that the attributes of a concept apply to its descendants without exception. Thus, the proper location of any concept within the taxonomy can be uniquely determined from the concept's definition by an automatic process known as classification. (3) They restrict the expressiveness of their language to achieve relatively efficient performance.

We first survey important general issues in the field of knowledge representation, consider the semantics of concepts and their interrelationship, and examine the intertwined notions of taxonomy and inheritance. After discussing classification, we present a number of implemented terminological systems in detail, along with several hybrid systems which couple terminological and assertional reasoning components. We conclude by assessing the current state of the art in terminological knowledge representation.

1. Introduction

A major focus of contemporary work in knowledge representation is on domain-independent knowledge base services which act as repositories for the knowledge used in artificial intelligence (AI) systems. Such utilities support storage and retrieval of domain knowledge coupled with limited generic inferencing capabilities. Ideally, they make it easy for a user to encode knowledge and for an AI system to use it. An important class of knowledge addressed by many of these systems is the definition of conceptual terms used in some domain and reasoning with those terms.

In this paper, we survey issues in the field of semantic network and frame-based knowledge representation formalisms, provide an overview of an interesting subclass of systems known as terminological logics which are based on definitional taxonomies[1]of *objects* or *concepts* or *frames* and also examine some reasoning systems built in association with them. All semantic network and frame formalisms organize a system's knowledge in terms of nodes and links among the nodes, but the internal structure of nodes and links varies significantly from one approach to another. Moreover, as we shall see, there is substantial disagreement over the proper way to interpret nodes and links. Unfortunately, even the terminology of the field has not been consistent. For example, the notions of frames and semantic networks, at one time considered distinct ideas, have by now merged in common usage.

We concentrate on formalisms where conceptual primacy is accorded to the nodes, and refer to them in general as *object-centered* knowledge representations[2]. We use the term *network* pervasively when referring to the collection of nodes and links comprising any given representation, without implying anything else about their structure. When we discuss some particular work, we endeavor to retain the author's own terminology.

This paper aims to answer the following questions about object-centered knowledge representations with definitional taxonomies:

- What are they?
- How do they relate to other knowledge representation formalisms?
- When are they most appropriate choice?
- What have they been used for in AI?
- What are the problems with and issues surrounding these formalisms?
- How have these problems and issues been addressed to date?
- What are the current research directions and, especially, how are such representations being integrated with generic reasoning components on a principled basis?

[1]Meaning that assertions about a concept apply to its descendants in the taxonomy without exception (more about this later).

[2]In [Nilsson 80], an object-centered representation is defined as ‘‘a representational scheme in which the facts are indexed by terms denoting entities or objects of the domain.’’

1.1 Knowledge Representation and Knowledge Bases

For the purposes of this survey, a *knowledge base* is considered to be an explicit, largely declarative collection of knowledge organized for use by an artificial intelligence system. A declarative knowledge base must be paired with an interpreter to make it useful. In addition to direct retrieval from the knowledge base, the interpreter generally makes inferences from its contents. Together, the knowledge base and interpreter form a *knowledge base system* which serves as a utility for AI applications. Although an ideal interpreter would provide users with the explicit contents of the knowledge base as well as all of its implicit logical entailments, today's interpreters provide incomplete support for the latter.

A domain independent *knowledge representation formalism* is used for expressing the information content of a knowledge base. It specifies a notation for recording knowledge, a denotation for interpreting it, and an implementation. A *knowledge representation* aspires to be more than a data structure by the generality of its expressive power and (one hopes) by its systematic, implementation independent semantics. The need for systematic denotational semantics in knowledge representation is argued in [McDermott 78]. A particularly salient feature of network representations is the organization and indexing of concepts by *semantic* content in correspondence with the world being modeled.

1.2 Background: The Case for Symbolic Representation

We now sketch the philosophical underpinnings of symbolic knowledge representation in AI. The Physical Symbol System Hypothesis [Newell and Simon 76] asserts that knowledge can be represented symbolically, that intelligence consists of manipulating symbols in a suitable manner, that systems which perform intelligent symbol manipulation can be realized in the physical universe and indeed that human beings are living examples of such systems. The Physical Symbol System Hypothesis lends importance to the knowledge representation enterprise and underlies most of the work in the field to date. Assuming that this hypothesis is true, it follows that computers can, in principle, be programmed to exhibit intelligent behavior. The prospects for successful realization of such artificial intelligence have been the subject of lively debate. This paper is presented from an artificial intelligence perspective, and in what follows we will not question the physical symbol system hypothesis.

The Knowledge Representation Hypothesis proposed by Smith [Smith 85] rests on the assumptions of the physical symbol system hypothesis, and additionally claims that:

> ''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 semantic attribution, plays a formal but causal and essential role in engendering the behavior that manifests that knowledge.''

Barr has argued that a simple *representational correspondence* of this type is inadequate to account for cognition, particularly when it comes to metacognitive issues such as the *tip-of-the-tongue* phenomenon and the *knowing not*[3] phenomenon [Barr 80]. Still, it seems fair to conclude that a system reasons with symbolic knowledge if and only if it operates in

[3] Awareness of things that you don't know.

accordance with the knowledge representation hypothesis. We shall also take it for granted that the knowledge representation hypothesis is plausible.

Newell has been concerned with the question ‘‘What is knowledge?’’ and in [Newell 82] he focuses on formal treatment of the term *knowledge* as distinct from *representation* in artificial intelligence. Newell argues for a level of representation which he terms the *knowledge level* apart from, and above, the symbol level. The knowledge level concerns itself with what is represented, i.e., semantic content, rather than how it is represented in terms of symbolic structures. Hence, a knowledge level view is useful in exploring what is possible in symbolic knowledge representation. The knowledge level itself is composed of goals, actions and bodies. At this level, the system is an agent. The agent decides what actions to take by processing its knowledge in a rational manner to attain its goals. Thus, intentionality is a crucial factor in Newell’s mind. Newell states the Knowledge Level Hypothesis as follows [Newell 82]:

> There exists a distinct computer systems level, lying immediately above the symbol level, which is characterized by knowledge as the medium and the principle of rationality as the law of behavior.

KRYPTON is one knowledge representation system which deliberately provides a knowledge level account of its services via implementation-independent semantics. We will cover KRYPTON in section 7.3.

1.3 A Bit of Heritage

The terminological networks we concentrate on in this paper follow in the tradition of AI research on semantic networks and frames. For the sake of history, we touch upon the seminal contributions of Quillian and Minsky, who are credited with originating semantic nets and frames, respectively [Quillian 68, Minsky 75].

Early work on semantic networks by Quillian was intended to provide a psychological model for human associative memory. He represented dictionary information about words and their associations as nodes interconnected by links. His system tried to compare and contrast pairs of words. Quillian developed the idea of *spreading activation* to model the process of association, or discovering relationships, between concepts.

Frames were proposed by Minsky in his landmark paper as ‘‘a data-structure for representing a stereotyped situation, like being in a certain type of living room, or going to a child’s birthday party.’’ Minsky’s frames were networks of nodes and relations intended to be invoked from memory to provide a framework for interpreting a new situation or viewing a situation from a different perspective.

In the past, frame systems were somewhat distinguished from semantic networks by their emphasis on default reasoning and on the internal structure of frames. As we remarked earlier, semantic network systems and frame systems have evolved and proliferated to the point that there is no longer any clear distinction between the two.

We mention in passing that two notable early attempts at building object-centered knowledge representation systems were FRL [Roberts and Goldstein 77] and KRL [Bobrow and

Winograd 77]. These languages were extremely powerful, but their semantics were unclear.[4]

1.4 Basic Ideas of Object-Centered Representation

In general, an object-centered knowledge representation is intended to *denote* a collection of entities in some external world and their interrelationship. External entities exhibit a set of attributes and corresponding restrictions on the values these attributes may have. These entities are *modeled* in the representation by objects, or concepts, which contain structure in the form of slots (or roles) that model the concept's attributes. The slots, in turn, have fillers to model the corresponding attribute values. In other words, a mapping exists between the objects and slots internal to the representation and the entities and attributes external to it. Nevertheless, it is essential to remember that the interpretation of objects as models of (real) world entities is largely in the mind of the system designer. Drew McDermott's highly entertaining critique, ''Artificial Intelligence meets Natural Stupidity'', makes this point cogently as it assails the presumptuous vocabulary that is often adopted by AI researchers to describe their systems [McDermott 76]. McDermott's admonition to maintain a realistic perspective on the meaning of one's representation is still worth remembering.

Since the fillers of slots can themselves be objects, a collection of objects may naturally be seen as a labeled directed graph. In this view, the objects are nodes and directed arcs connect objects with those objects which fill their slots. A simple graphical presentation of a fragmental object-centered representation is shown in Figure 1-1.

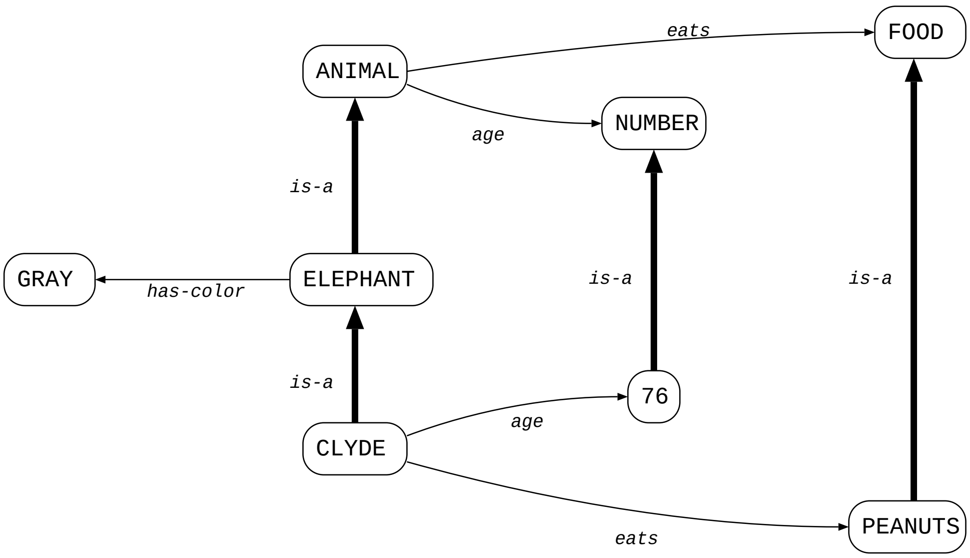

Figure 1-1: Sample Taxonomy

As Figure 1-1 suggests, some kinds of knowledge are straightforward and natural to represent with the object-centered approach:

---

[4]Lehnert and Wilks have written a criticism of KRL, which is now regarded as an ambitious failure [Lehnert and Wilks 79]; a rejoinder on the part of the KRL designers is [Bobrow and Winograd 79].

- **Individuals,** e.g., Clyde, the number 76.
- **Classes,** e.g., Elephants, Animals, Food, Numbers.
- **Properties,** e.g., Elephants are grey.
- **Relationships,** e.g., Clyde is an elephant. Elephants are animals. Animals eat food. Clyde eats peanuts.

Concepts are almost invariably organized into a taxonomy, or generalization hierarchy, via a distinguished generalization link commonly known as *is-a*. With respect to the taxonomy, one may say that a concept generalizes its descendants and that it specializes its ancestors. In other words, as one follows is-a links upwards in the taxonomy, one encounters increasingly general concepts. In the opposite direction, one encounters increasingly specific concepts. The root of the taxonomy is a completely general concept which subsumes all other concepts.

Object-centered representations often distinguish between *generic* concepts and *individual* concepts. A typical interpretation would hold that a generic concept denotes a set of entities and that an individual concept denotes one particular element of the set. With such representations, it is useful to have two kinds of is-a links. The first connects a generic concept to the most specific generic concept(s) that subsume it, while the second connects an individual concept to the most specific generic concept(s) that subsume it. To distinguish between these is-a links, the latter is typically referred to as an *instance-of* link.

Numerous artificial intelligence textbooks introduce the ideas we have just outlined. Some examples are [Nilsson 80, Rich, E. 83, Winston 84, Barr and Davidson 81]. Finin's tutorial is brief and highly readable [Finin 86a]. The paper by Fikes and Kehler describes more recent ideas with emphasis on robust commercial systems such as KEE, which the authors developed at Intellicorp [Fikes and Kehler 85]. Basic techniques for implementation are covered in [Finin 86b] and [Charniak, Riesbeck and McDermott 87]. A good overview paper on knowledge representation and reasoning is [Levesque 86a]. A collection of notable research papers is [Brachman and Levesque 85].

1.5 Motivation for the Object-Centered Approach

The *raison d'etre* for any knowledge representation is to support storage and retrieval of knowledge as well as reasoning with knowledge, often for the purpose of problem solving. This raises the question: How can we use object-centered knowledge representation to support these ends? Several suggestions have been made, but bear in mind that there has been controversy over the validity of some intended uses and whether the object-centered approach offers any unique contributions. In the remainder of this section, we discuss several positive features that have been claimed for the object-centered approach.

1.5.1 Organization

A concept can bundle related information together in a unit. This makes it easy to process related information as a group. In addition, the entire concept can be given a name for convenient reference, and concepts are often named so as to evoke the entities they represent. For instance, a concept denoting **elephant** may indicate that elephants are mammals, that they are grey, that they eat peanuts, and so on. In general, it is expected that slots will be defined to express just those conceptual relations likely to assist users of the knowledge representation service.

A taxonomy clarifies the relationship among concepts. It allows a concept to be defined as a specialization of one or more parent concepts in the taxonomy. For example, **elephant** is a specialization of **mammal**. Moreover, collections of concepts may be arranged in several hierarchies simultaneously. For instance, parts hierarchies, consisting of part/subpart relationships, may coexist with the is-a hierarchy. A taxonomy assists with retrieval of concepts which match some description by providing a basis for search. Likewise, a taxonomy supports inference about the relationships among concepts.

1.5.2 Instantiation and Type Checking

The definition of a generic concept constrains the definition of a subsumed individual concept. An object which describes a generic concept can serve as a template for individual concepts which comprise its instances, i.e., generic concepts provide guidance for the specification of individuals. The act of creating an individual concept which conforms to a generic concept's definition is called *instantiation*. For example, if the generic **person** concept states that a person's age is a positive number, then 25 is an acceptable value for an individual person's age, whereas the color red is unacceptable. We can also create a generic concept for **child** as a specialization of person which further restricts children's ages to be less than 18.

Whether or not generic concepts play an active role in instantiation, they can support type checking; a concept can be tested against the definition of one or more generic concepts which it should conform to. A generic concept may be tested against another, more general, generic concept. For instance, if we define children to be people, we would expect the system to report an error if we attempted to define the age of children to be a color. Likewise an individual concept may be tested against a generic concept. Assuming that Nancy is a particular child, we would also expect an error report if we tried to define her age as red.

1.5.3 Defaults

A generic concept may provide default values for attributes of the individual concepts it describes. Suppose we choose to assume that the color of an elephant is grey in the absence of contradictory information. To implement this, the color role of the generic elephant concept may be annotated with a default value indicating grey. If we then create an individual concept describing a particular elephant named Clyde without specifying Clyde's color, the default mechanism provides a means for assuming that Clyde is grey. The value restriction of the generic elephant concept's color role may simply be **color**; this still permits us to describe individual pink elephants and white elephants without introducing a contradiction.

1.5.4 Criteriality and Matching

If we interpret the definition of a generic concept to provide sufficient conditions for class membership, then any individual concept which conforms to the generic's definition must be a member of that class. This has been called the *criteriality* inference. For example, the definition that women are both human and female provides sufficient conditions for ascribing womanhood to an individual -- knowing only that Liz is human and female, we may conclude that she is a woman. As it happens, this definition provides necessary conditions for womanhood as well -- anything which is not human, or is not female, is certainly not a woman.

Even if the definition of a generic concept does not supply sufficient conditions for class membership, it may still provide necessary conditions. We may match an individual concept against such a generic to determine if their definitions are consistent. If so, it is plausible that the individual concept is a member of that generic class. For example, someone with a Ph.D. degree who occupies a private office is plausibly a professor; however, she might instead be a research associate.

1.6 Object-Centered Representation Versus Formal Logic

There has been considerable debate over the relationship between the object-centered approach and formal logic, usually taken in this context as first order predicate calculus (FOPC). This controversy has been clouded, historically, by lack of agreement regarding both the purpose of object-centered representations and their semantic import. This problem has been compounded by lack of precision in defining particular object-centered formalisms. On the other hand, it is not exactly clear what the logic imperialists[5] stand for either. Israel and Brachman, who draw a distinction between a *language* and the formalism which goes with it [Israel and Brachman 81], voice this sentiment with a flippant answer to their own question:

Q: "... what <u>does</u> one embrace when one accepts a 'standard' quantificational formalism?"

A: "Darned if we know ..."

They make the point that besides a language, a formalism also encompasses a calculus and its implementation.

First order predicate calculus has certain desirable qualities for knowledge representation, chiefly its broad expressive power coupled with its well-defined semantics and rules of inference. Note that although numerous extensions to FOPC have been proposed, including modal logics, temporal logics, higher order logics, etc., there is little agreement on these extensions. Moore has argued that in some sense, logic and deduction are the only way to deal with the incomplete knowledge of a situation that is found in many problems [Moore 82]. Conversely, predicate calculus, *per se*, has significant shortcomings with respect to notational efficiency and computational effectiveness. Inefficient inference mechanisms are the direct result of predicate calculus' generality. Moreover, predicate calculus has no facility for aggregation of information.

Woods espoused the view that ‘‘concepts are more than predicates’’ and illustrated his assertion with the well-known blocks world arch example, shown in KL-ONE notation in Figure 1-2 (The details of the notation are not essential in this context). The representation shows that an arch is composed of three blocks, one being a lintel and two being uprights, and specifies certain constraints on the physical relationships among them. What predicate is the concept of an arch identified with? Woods suggests several possibilities:

- **ARCH(X, Y, Z):** A predicate on three blocks which characterizes their relationship

---

[5]Tongue in cheek term due to [Israel and Brachman 81].

such that they constitute an arch.

- **ARCH'(X):** A single argument predicate which is true of an individual arch.
- **ARCH''(S):** A predicate on a situation S which is true if some arch exists in that situation.

Woods maintains that the arch concept is an abstraction that ties together all of these related predicates, and others, which jointly characterize what an arch is.

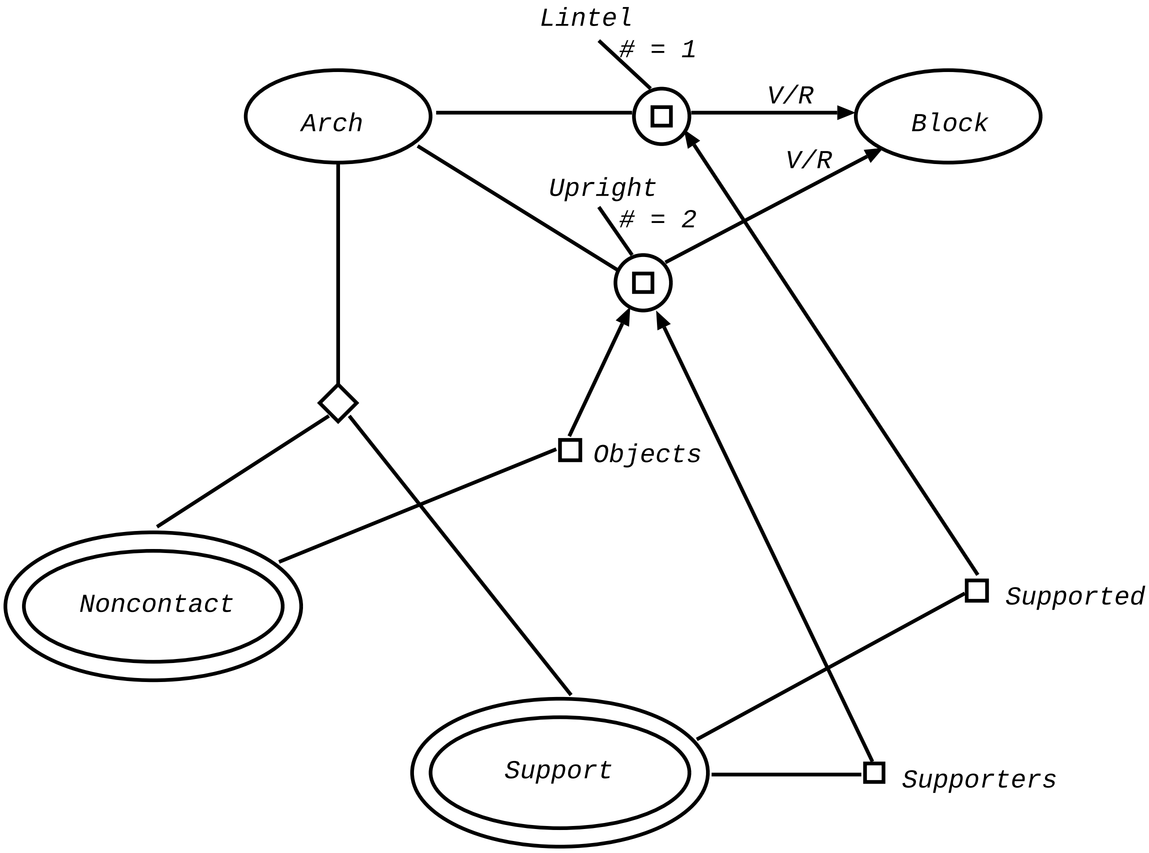

Figure 1-2: Blocks World Arch [Woods 86]

In large measure, the debate comes down to whether or not, as Hayes contends, object-centered representations are nothing more than an implementation of logic, and as such, have little of value to offer [Hayes 77, Hayes 79]. Hayes argues that the idea of frames, in particular, amounts to storing groups of logical assertions as named units and supporting retrieval of said units by indexing their names. It seems that various object-centered formalisms *can* be axiomatized by some logic, be it standard or otherwise. Even so, the fact remains that logic must be implemented before it can be used in a computer system. We maintain that network implementations provide useful tools for partitioning, organizing and using knowledge in an efficient, natural and readable manner.

1.7 A Knowledge Base *is-a* Database ... and More

Obviously, the knowledge base systems treated in this paper have much in common with traditional database systems. While only one recent paper, [Borgida *et. al.* 89], explicitly presents such a system from a database perspective, casting CLASSIC in the guise of a data model, the analogy is clear and worth detailing in terms of database terminology. Later on, we see how knowledge base systems differ from traditional database systems by providing enhanced representational power at the expense of performance.

Operators for describing complex phenomena such as concepts (and roles) comprise a *data definition language* and the descriptions formed for a particular application constitute a *schema*. Operators for accessing aspects of generic concept definitions provide the services of a *data dictionary*. Role value restrictions and role constraints[6] which specify relationships among the values of a concept's roles both serve to enforce *integrity constraints*.

Assertions about individual concepts are the information content of the database. Functions to add, delete or modify descriptions of individuals amount to a *data manipulation language*. Functions to ask about individuals form the basis for a *query language*, which may or may not be provided as an integrated utility by a particular system.

Knowledge bases differ from traditional databases in several crucial ways, as discussed in [Brachman 88]:

- Knowledge bases are capable of more powerful expression. In particular, they must deal explicitly with incomplete information, with exceptions, and with rich relationships among generic concepts.
- Knowledge bases treat their content at a semantic level, i.e., as knowledge rather than as data. In this vein, they may attempt to account for the semantics of natural language.
- The emphasis is on support for interpretation of knowledge and inference with knowledge, rather than simple data retrieval. A knowledge base system may be expected to take an active role by providing information implicit in its contents. For example, it may reason about categorical relationships among the concepts it represents and automatically place concepts in useful relation to one another.
- In contrast with data bases, structures which define generic concepts comprise a large portion of a knowledge base, perhaps even a larger portion than their instances. That is, knowledge bases tend to emphasize intensional definitions while traditional databases are geared towards extensional information. Furthermore, generic concepts have rich interrelationships and tend to change fairly often.

1.8 A Road Map for the Rest of This Paper

In the following section we set forth a range of knowledge representation issues in greater detail. In section 3, we take a careful look at the semantics of nodes and links themselves and discover the many difficult issues involved. Section 4 is devoted to the intertwined notions of taxonomy and inheritance. Here again there are numerous potential pitfalls and a variety of

---

[6] Or role value maps.

proposed solutions. Subsequently, Section 5 examines classification, a process for automatically computing the proper location of a concept in a definitional taxonomy. Then in section 6 we turn our focus to particular object-centered knowledge representation systems. We have chosen several examples which are historically important or which reflect the current state of the art among terminological knowledge representation systems. We proceed in section 7 to explore some systems which strive to ameliorate the shortcomings of object-centered representations by synthesizing them with complementary approaches such as rule-based systems. For the sake of contrast, section 8 presents some alternative systems that represent knowledge and reason with frames. Finally, we conclude with a general assessment of the work and indicate some current research directions.

2. Knowledge Representation Issues

Is a knowledge representation formalism adequate to capture the particular knowledge we want it to express? It is difficult to characterize and compare object-centered formalisms because they have been used for many different purposes and interpreted in widely varying ways. Nevertheless, we present a sometimes overlapping range of issues confronting the field of knowledge representation and attempt to show how these issues relate to the object-centered approach. In the sections that follow we contrast the advantages offered by the object-centered approach with the disadvantages. We will be concerned with what can be represented in an object-centered formalism as well as how easily we can represent things. Brachman holds that the key challenge before the field is to isolate suitable primitives for a representation so that it achieves ‘‘semantic coherence’’ [Brachman 79]. We turn first to this issue.

2.1 Semantic Coherence

Historically, object-centered representations have suffered from imprecise or even careless definition. Each representation provides some set of primitives. In the past, the primitives have often been chosen on an *ad hoc* basis. Brachman has attempted to tame the resulting confusion by proposing specific levels at which to view sets of primitives. He identifies five successive levels, each of which has a characteristic set of link types:

- **implementational level:** At the lowest level, a network may be conceived simply as an implementation of some logical language. As such, it is just a data structure, composed of atoms and pointers.
- **logical level:** Here, the components of a network can be thought of as logical primitives, where nodes are predicates and propositions, and links are logical connectives. Importantly, a network at the logical level serves to index its logical content by grouping related components together in graphic proximity.
- **epistemological level:** The identification of a distinct epistemological level was a central contribution of Brachman’s work. As he says, ‘‘The formal structure of conceptual units and their interrelationships as *conceptual units* (independent of any knowledge expressed therein) forms what could be called an *epistemology*.’’ An epistemological level network is made up of nodes representing concepts, using an explicit set of concept types provided by the formalism. Nodes may also be composed of conceptual subpieces. The links denote inheritance and other structuring relations. We will see just what epistemological primitives Brachman had in mind when we examine his KL-ONE system in section 6.2.
- **conceptual level:** Much of the work in object-centered representation has been inspired by the requirements of natural language understanding, where networks are used to capture the intrinsic meaning of linguistic expressions. Here, at the conceptual level, the representation is supposedly language independent. A small set of network primitives with semantic import, such as primitive entities and actions, or perhaps case relationships, are chosen. Some researchers, principally Schank and his followers, further strive to identify a *minimal* core of primitives sufficient to capture the semantic content of natural language. Schank’s conceptual dependency theory, known as CD, is prototypical work of this sort [Schank 72]. The contributions of Schank and his followers are thought-provoking, but as we are focusing on the form of a knowledge representation rather than its content, the selection of core conceptual primitives is beyond the scope of this paper, and we

will not consider it further.

- **linguistic level:** A network of linguistic primitives can be geared towards a specific language such as English. This highest level has hardly been explored. One example system, for sake of reference, is OWL [Szlovits *et al.* 77].

How is Brachman's stratification useful? He identified three criteria for assessing a network's coherence at any level: *neutrality*, *adequacy* and *semantics*. First, a network at one level should be neutral to higher levels in that it does not constrain higher level design choices. Second, a network should provide adequate support for the next higher level. Lastly, the semantics of a chosen level should be clear and precise. Ideally, one would want formal definitions of nodes, links and operations on them.

The fundamental design principle of *layering* has found wide applicability in computer science. Operating system architecture is one case in point where a great deal of clarity and leverage have purportedly been achieved [Tanenbaum 87]. The initial version of KL-ONE had a layered *implementation* as did Rich's CAKE system [Brachman and Schmolze 85, Rich, C. 82, Rich, C. 85]. I am not aware of any other object-centered KR system that actually used a layered implementation.

While it is important for a representation system to achieve coherence at a particular level of representation, meeting this goal is no easy matter. Brachman contends that difficulties with earlier formalisms were due, in part, to the fact that their primitives were not chosen from a single level. His pioneering work on KL-ONE and subsequent systems described below, e.g., [Brachman *et al.* 83a, Brachman *et al.* 83b, Brachman and Schmolze 85, Borgida *et. al.* 89], consciously addressed this situation by concentrating on the epistemological level. This work has had profound influence on object-centered knowledge representation. With carefully restrained design, object-centered representations can have perfectly coherent semantics. Nonetheless, designers of commercial products such as KEE and ART have chosen to emphasize power over semantic coherence with an *ad hoc* collection of tools, perhaps because they feel compelled to compete with each other in terms of ''features''. Such systems are only defined operationally, and no formal account of their semantics is available.

2.2 Logical Power

Can a representation express the full range of logical relationships? Can a system make sound and complete inferences? Observe that there exists a fundamental tradeoff between logical power and computational efficiency [Levesque and Brachman 85]. It is not presently known how to construct reasonably efficient inference mechanisms which support the full generality of first order logic. Thus, the real challenge, as we will see, is to strike an appropriate balance. Nevertheless, since first order logic sets the standard for formal representations, it is important to consider how well object-centered representations are able to express the various constructs of first order logic. We will look at conjunction, disjunction, negation, universal quantification and existential quantification. We focus our discussion by considering a simple representation consisting only of taxonomically organized concepts which are further interconnected by means of roles.

Conjunction does not pose any problem. A concept may be defined as the conjunction of some other concepts by simply placing it underneath each of them in the taxonomy simultaneously. Consider Figure 2-1 where a woman is defined as both female and human. Similarly, a role may convey a conjunction by filling it with a conjoined concept. In Figure 2-1, the mother of a baby is defined to be a woman.

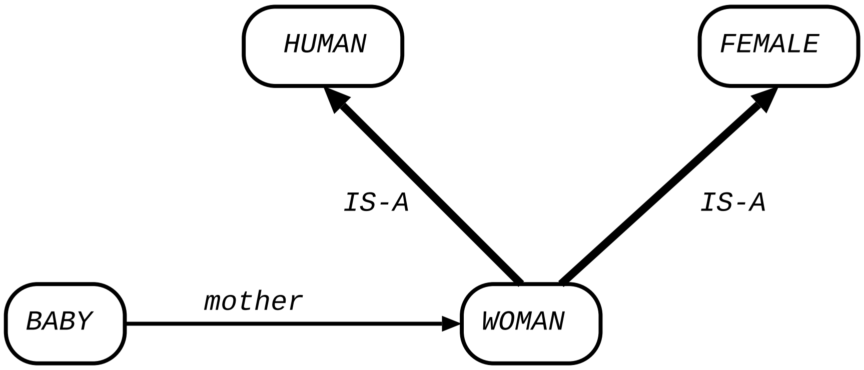

Figure 2-1: Conjoined Concept

If two or more of the concepts being conjoined have a particular role in common, but the associated value restrictions are not identical, then the value restriction of the corresponding role of the conjoined concept must reflect their logical conjunction. Needless to say, the value restrictions being conjoined should not be contradictory. Consider Figure 2-2. Some systems will automatically create a suitable conjoined concept to fill the role if it does not already exist. This might surprise the user, especially if the system creates it without notice.

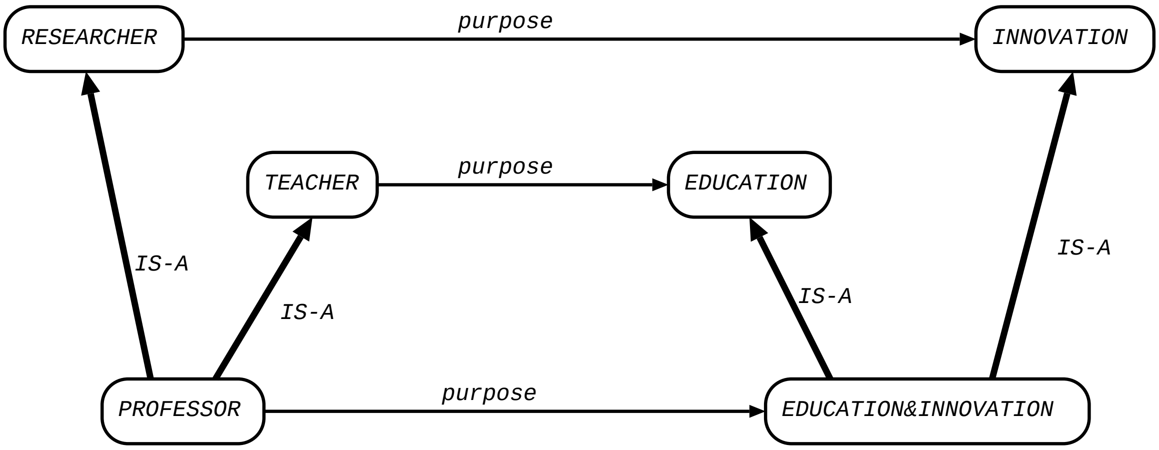

Figure 2-2: Conjoined Concept with a Specialized Role

Disjunction is a problem for the basic concept taxonomy. To see this, consider placing a concept for **cat-or-dog** in relation to the concepts for **cat** and for **dog** in a taxonomy. The difficulty is that a link to a parent conveys specialization and multiple parents are defined to be interpreted conjunctively, so there is no proper place in the taxonomy for a concept which is the disjunction of some other concepts. An additional mechanism would be required to specify the disjunction operator and delineate its operands.

Negation is also a problem for the basic concept taxonomy. Concepts for ***x*** and for ***not X*** would require a common ancestor. Clearly one could neither subsume nor be subsumed by one another. But more would be needed to capture their complementary relationship and the fact that together they are exhaustive.[7]

Simple universal quantification is implicit in the association of a role with a generic concept; unless otherwise indicated, the value of that role applies to all subordinate concepts, both individuals and generics. The definitional taxonomy systems we examine in this paper do not permit exceptions to such universal statements. This is a significant limitation.

Existential quantification is not supported by the basic conceptual taxonomy, e.g., it is not possible to assert the existence of an unspecified individual for some generic concept. Still, role value restrictions can be seen as existential variables, e.g., the **animal** concept's **eats** role with a value restriction of **food** *may* be taken to mean that for every animal, there exists some food which it eats. General nested quantification is also unsupported.

Contemporary object-centered representations based on definitional taxonomies do not support disjunction, negation or existential quantification. Some of these systems simply do not require such operations for their intended purpose, which is to define a conceptual taxonomy as a ''terminological'' resource. In general, such systems rely on efficient, automatic computation of taxonomic relationships among concepts in order to maintain the definitional nature of the taxonomy. Disjunction, negation and general quantification would complicate these calculations enormously. Instead, systems with definitional taxonomies leave it up to associated ''assertional'' reasoning components to provide these facilities in support of problem solving.

A separate class of efforts have extended semantic networks to encompass the full power of predicate logic. An early example of a semantic network with the full power of first-order logic was Shapiro's SNePS system (Shapiro79). Schubert also produced a semantic network formalism with a clear and deliberate correspondence to predicate calculus. His system included modal operators as well as definite and indefinite references (Schubert76). Deliyanni and Kowalski defined an extended semantic network formalism as a syntactic variant of clausal logic (Deliyanni-Kowalski79). The clausal interpretation of networks allows them to support both top-down and bottom-up reasoning as special cases of resolution. The best known work in this area is Hendrix's *partitioned networks* [Hendrix 79]. Hendrix partitions the network into *spaces* of related nodes and arcs which can be treated as a unit. For instance, to handle quantification, spaces correspond to the scope of variables.

2.3 Logical Consistency

It is desirable for a knowledge base to be internally consistent. Formal logic provides inference rules which are known to be valid. When they are applied to true facts, any new facts which result must also be true. Similarly, it is always possible to see if a new fact is consistent with existing information: just add it to the knowledge base and see if a contradiction results

---

[7]NIKL, for example, has **disjoint** and **cover** constructs. While these *might* be used together to partition subconcepts and thus express negation in an indirect fashion, NIKL doesn't make such inferences.

when rules of inference are applied.[8] Since the inference rules are known to be valid, any contradiction must be due to inconsistent facts. The importance of internal consistency is not universally accepted. Minsky has argued that it is not necessary and perhaps undesirable for an intelligent system undergoing development [Minsky 75]. Rather, he emphasizes the importance of handling possible inconsistencies and learning from mistakes.

Present day object-centered representations, other than strict encodings of formal logic, are somewhat lacking in formal definition. Unfortunately, they cannot offer the same guarantee of consistency because they do not come with certifiably valid inference rules. Rather, the semantics of a network depends on the interpreter which processes it. The performance of the interpreter is not always entirely documented or even understood.

Relatively recent systems in the tradition of KL-ONE have begun to address this problem. The KL-ONE formalism was meticulously designed and its enforced semantics enable use of an automatic classification algorithm to maintain the taxonomy, resulting in a high degree of consistency.

2.4 Incomplete Knowledge

According to Woods, a system should be able to remain noncommittal in the face of incomplete knowledge [Woods 86]. This ability is directly related to the system's logical power. First order logic excels at the expression of incomplete knowledge by means of negation, disjunction and quantification. We will provide an example of each.

- **Negation:**
  $\neg\, bird(Clyde)$. But what *is* Clyde?
- **Disjunction:**
  $elephant(Clyde) \vee bank\text{–}robber(Clyde)$. But which one? Or both?
- **Existential Quantification:**
  $\exists x\ eats(Clyde, x)$. But what in particular does he eat?
- **Universal Quantification:**
  $\forall x\ eats(x) \rightarrow animal(x)$. Is there anything that eats? And if so, which things are eaten?

Most structured object representations cannot express incomplete knowledge like this. Of course, semantic networks that have been extended to provide the full power of first order logic can, e.g., [Hendrix 79]. The tradeoff, again, is between logical power and computational efficiency.

2.5 Semantic Coverage

There are many kinds of knowledge that we should be able to express in an ideal representation. As we have seen, the object-centered approach lends itself to constructing a taxonomy of concepts, attributing properties to concepts and establishing binary relationships between concepts. More specialized forms of knowledge pose a stiffer challenge. Varieties of

---

[8] In the worst case, this may take quite some time, however.

specialized knowledge which seem ubiquitous include:

- Temporal relationships
- Spatial relationships
- Part-of relationships
- Numbers
- Intervals
- Case-based reasoning, as in medicine or the law
- Meta-knowledge, such as the notions of belief and degree of certainty
- Reflection, or reasoning with self description

It would be useful for knowledge representation systems to come equipped with specialized facilities, or *built-in semantics*, for reasoning with specialized knowledge. For the sake of illustration, consider a knowledge representation which addresses the decomposition of parts into subparts. One would like to have parts hierarchies built around the *part-of* relation which are orthogonal to the is-a hierarchy. Reasoning with interwoven parts hierarchies and is-a hierarchies *and* their interaction is a challenging matter. For example, Schubert observes that confirming the part-of relationship among two nodes in a parts graph is NP-complete and he proposes a way to organize parts graphs for efficient information retrieval [Schubert 79]. Another difficult problem studied by Schubert is how to inherit part-of relationships in an is-a hierarchy. A second example, the Researcher system developed at Columbia University, had a frame-based representation for representing complex physical objects in terms of (1) a parts hierarchy, (2) physical and functional interpart relations and (3) properties of the parts. Researcher was among other things a learning system which autonomously organized examples of physical devices into a generalization hierarchy [Lebowitz 85].

Recent classification-based systems such as K-Rep and Loom provide notations for expressing specialized concepts such as numbers, sets, and arbitrary intervals on the number line or the time line. They can automatically compute subsumption relations between such concepts.

2.6 Domain Modeling

Generally speaking, a knowledge base should support mappings from world states to knowledge base states, from world events to knowledge base operations, and from mental states to knowledge base states. See Figures 2-3, 2-4, and 2-5 adapted from [Mays 88]. Viewing representations in such a framework suggests several questions raised by Bobrow [Bobrow 75]:

- **Domain and Range:**
  What is being represented?
  How do objects and relationships in the world correspond to units and relations in the model?

- **Operational Correspondence:**
  In what ways do the operations in the representation correspond to actions in the world?

- **Process of Mapping:**
  How can knowledge in the system be used in the process of mapping?

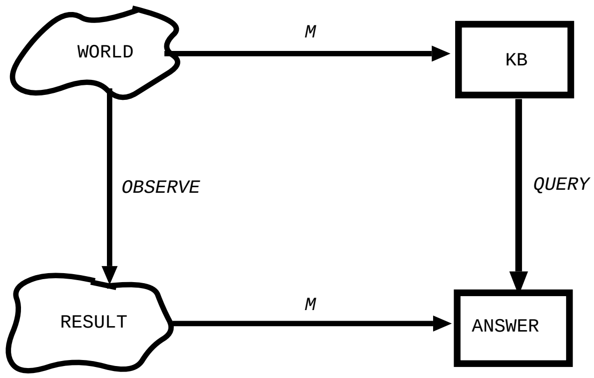

Figure 2-3: Mapping World States to KB States [Mays 88]

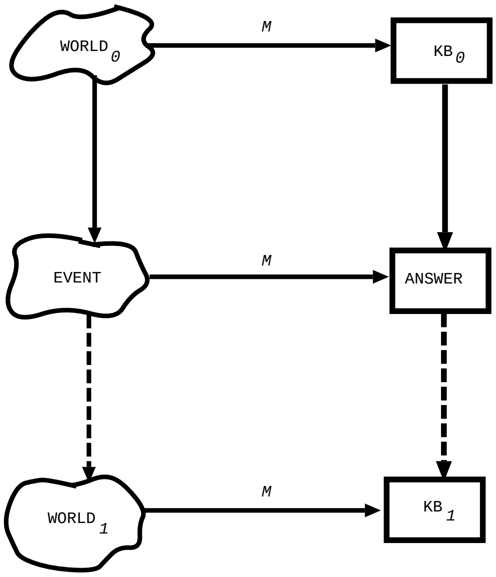

Figure 2-4: Mapping World Events to KB Updates [Mays 88]

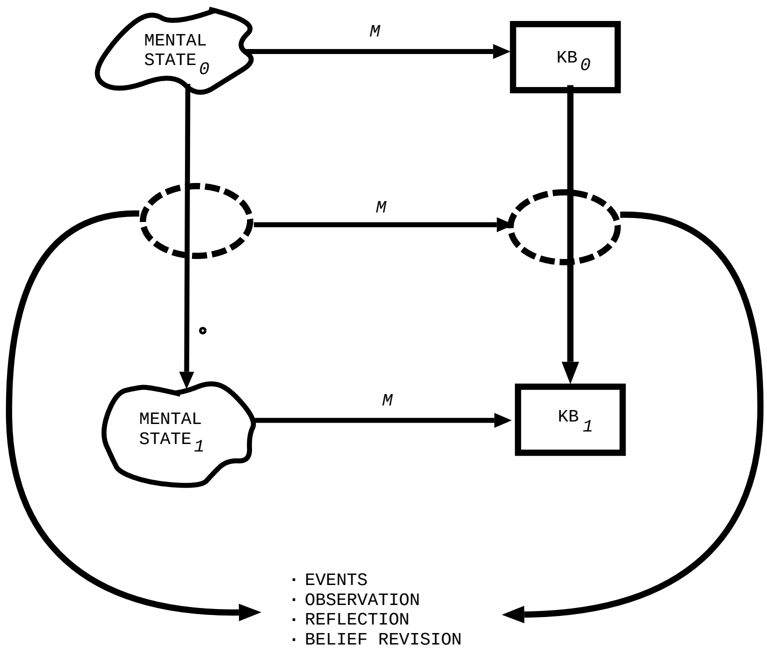

Figure 2-5: Mapping Mental States to KB States [Mays 88]

2.7 Vividness

Hector Levesque has advanced the notion of *vivid* knowledge bases to support *model-based* reasoning [Levesque 86b]. A vivid knowledge base is an isomorphic model: the contents are structured by analogy to whatever they represent. One would also manipulate the model by analogy to manipulation of entities in the domain. Initial work in this area has proven difficult. An architecture for a vivid knowledge base system and some preliminary ideas about how to construct a vivid knowledge base as a complete database of ground, atomic facts from incomplete facts in a more expressive language are given in [Etherington *et al.* 89]. They refer to this process as *vivification*. We can say that object-centered representations are relatively vivid compared to unstructured sentences in logic.

2.8 Expressive Precision

Knowledge should be available to problem solvers at an appropriate level of detail, or *granularity*. Bobrow suggests that the level of detail should be consistent for a particular problem solving situation [Bobrow 75]. Woods notes that a representation should be able to convey essential differences while simultaneously capturing commonality among conceptualizations [Woods 86]. He observes that taxonomic organization with inheritance serves this purpose well. Distinctions can be achieved by specializing existing concepts into several variants. Likewise, generalizations can be realized by introducing general concepts that subsume several existing ones.

2.9 Domain Primitives

The expressiveness of a knowledge representation depends on the domain primitives which are chosen to construct a domain model. Most object-centered formalisms are domain independent, so the choice of domain primitives is up to the knowledge base builder. Our concern over domain primitives is limited to the requirement that the primitives supplied by a representation should allow us to model whatever domain primitives we find appropriate.

2.10 Notational Efficiency

The notation for a representation should be concise. At the same time it should offer conceptual clarity so that creation and extension of a knowledge base is straight-forward, even obvious. The notation should make it easy to reflect small knowledge changes in the knowledge base. When explicit knowledge is encoded in some representation, it is desirable that additional knowledge which it implies follows directly from the representation. Inheritance in object-centered representations does just this. With inheritance, attributive or procedural information can be explicitly attached to the most general concept in the taxonomy where it applies and implicitly shared with subordinate concepts. A notationally efficient representation can have positive consequences for computational efficiency.

2.11 Computational Efficiency

The computational tractability of a knowledge representation and its concomitant reasoning facility varies inversely with its expressive power. Thus a fundamental tradeoff exists. Levesque and Brachman have examined this tradeoff at the *knowledge level* (Newell82) and concluded that this tradeoff applies to all computational systems which reason automatically and correctly [Levesque and Brachman 85]. Indeed, they suggest that ‘‘much of the research in KR can be construed as trading off expressiveness in a representation language for a more tractable form of inference.’’

The decision problem in first-order logic is unsolvable, and even without quantifiers it is co-NP-complete[9] in the worst case. This may not be a problem for most applications in practice, but at the same time, no guarantees can be made. In [Levesque and Brachman 87] three ‘‘pseudosolutions’’ to the problem of intractability are discussed:

1. Speed computation with improved theorem proving methods, VLSI and parallel processing.
2. Relax the notion of correctness by returning a result after a given period of time, perhaps by answering unknown.[10]
3. Restrict the power of the knowledge representation language so that tractable performance is guaranteed.

Computational efficiency entails effective use of the twin resources, time and space. As we saw in the previous section, the tractability of a representation depends critically on the

[9]Considered to be unsolvable in polynomial time.

[10]Sometimes a default answer would be a reasonable alternative.

functionality it offers. Efficiency is also partly a matter of implementation. But from another point of view, the intrinsic organizational qualities of a representation have an important effect on the representation's efficiency. It is this last consideration which we now have in mind.

Problem solving programs need to *access* the knowledge base. In an object-centered formalism, the fact that knowledge about a concept is grouped together in a unit means that it is both natural and efficient for an interpreter to focus on the concept -- it can easily obtain whatever information is available about the concept without having to search throughout the knowledge base. Links are intended to make important relationships between concepts explicit. In fact, object-centered networks aspire to provide just the right associations required for effective reasoning. Hence an interpreter can effectively attend to a small group of related concepts. Using links, an object-centered formalism can often obviate extended searching through the knowledge base when following a chain of reasoning.

Problem solvers also need to *reason* with a knowledge base. Taxonomic organization supports reasoning at different levels of detail. Abstract reasoning with generic concepts can sometimes be far more effective than reasoning with an entire set of individual concepts.

Finally, problem solvers may need to *modify* the knowledge base according to information that they acquire or infer separately. This could mean adding or deleting concepts, or modifying existing ones. Here again taxonomic organization can prove useful. When problem solvers need to augment the knowledge base with hypotheses and inferences on a temporary basis, there should be efficient means to retract them *en masse*.

2.12 Data and Procedural Abstraction

If a knowledge representation system is to have well-founded semantics, it should present the knowledge base to the user as an *abstract data type* characterized by a set of well-defined operations. In keeping with *data abstraction*, details about the actual data structures used in the implementation should be hidden from users. The associated notion of *procedural abstraction* holds that details of the procedures which implement operations on an abstract data type should also be hidden. Together, declarative and procedural abstraction ensure that the user is isolated from the design of the implementation and any changes to it. Some of the consequences when such a strict interface to a knowledge base is not enforced are pointed out in [Brachman *et al.* 83b] and [Patel-Schneider 84]. When a user is able to access the actual data structures herself or modify the implementation of the operations, the knowledge representation can be manipulated in ways that were not intended by its designers. In fact, the user should not even be able to examine the implementation of the operations insofar as the mechanisms provide more information about the operations than the operation's definitions. In sum, the user should be precluded from undermining the implementation-independent nature of a representation's semantics.

2.13 Support for Knowledge Engineering

Knowledge engineering may be defined as the task of acquiring, representing and using knowledge appropriately. For the human knowledge engineer, a representation must be easy to understand and work with. Humans often think in terms of conceptual taxonomy. As a matter of fact, the initial work on semantic networks, begun by Quillian, was intended to provide a model of human associative memory [Quillian 68]. To the extent that this model is accurate, we can expect that humans will find object-centered representations natural.

An object-centered representation imparts organizational power to a knowledge base. This organization can be used to advantage by knowledge engineers who construct and maintain a knowledge base and by other people who access information from the knowledge base manually, as well as by computer programs which interact with it. We now discuss two aspects of knowledge representation utility from the perspective of human users: ease of access, and ease of acquisition and maintenance.

2.13.1 Ease of Access

How easily can a knowledge engineer examine knowledge? Knowledge engineers may wish to peruse a knowledge base by hand. The organizational qualities of object-centered representation aid knowledge base browsing in two ways. First, in just the sense that a structured object, with all its slots and fillers, represents a complex conceptual entity, the browser can perform manipulations in terms of complex conceptual entities. Second, the network structure of object-centered representations lends itself to presentation and manipulation via graphical browser software. Graphical proximity corresponds roughly to conceptual proximity. Further leverage may be gained from graphical browsers that are custom made to provide an interface for a particular object-centered representation. Although we will not study browsers in this paper, several of the systems described below have highly developed graphical interfaces.

2.13.2 Ease of Acquisition and Maintenance

How easily can a knowledge engineer create a new knowledge base? How well can she assimilate new information into an existing knowledge base? These questions address the general problem of knowledge acquisition which is pervasive in AI system building. When new information is added to a knowledge base or existing knowledge is updated, it is desirable to know how the change interacts with existing knowledge. Many times, the performance of problem solvers which access the knowledge base will be altered. That aside, a serious problem arises if new information contradicts existing information. We anticipate that knowledge acquisition will become an increasingly important challenge as AI systems grow larger in size and broader in scope.

The utility of taxonomic organization in knowledge acquisition is beyond doubt. It offers users a clear, well-defined framework for constructing a conceptual vocabulary and attributing information to concepts. Humans find it easy and natural to add new concepts by defining them as specializations of existing concepts, along with certain distinguishing features. Structured objects facilitate aggregation of information in a granularity which they find convenient and natural. They also ensure that a concept's description is localized in a single place. While assigning facts to frames is done by intuition, interactive knowledge base browsers can assist in this process.

Object-centered systems which support automatic placement of concepts into a taxonomy are useful during knowledge acquisition. By correctly locating a new concept in the taxonomy or relocating a modified concept, we see how its definition relates to those of other concepts.

Moreover, automatic classification can often detect contradictory definitions. Therefore, when concepts are treated as definitional, semantic consistency can be ensured by the system.

2.14 Support for Software Engineering

Software engineering concerns itself with the principled design and construction of large scale software. A non-trivial AI system can consist of several distinct problem solving components which tackle different aspects of a problem by very different means. A knowledge base system must serve their varied needs while allowing them to share information and exchange information on an opportunistic basis. Furthermore, individual components, and the system as a whole, will evolve over time. Woods has pointed out that a conceptual taxonomy is an excellent ‘‘coat rack’’ upon which to hang various procedures or methods. When problem solvers can be structured as collections of such procedures, a common object-centered representation provides the mechanism for their integration by serving as their interface. This methodology fosters a high degree of modularity and procedural independence.

2.15 Conclusion

We have now surveyed a ‘‘laundry list’’ of issues in knowledge representation which bear on the design of KR systems. Later on, when we evaluate particular systems, we will assess them in terms of their effectiveness in addressing these issues.

3. Semantics of Nodes and Links

Networks are composed of nodes connected by links. As such, they are nothing more than graphs. To be useful for knowledge representation, nodes and links must be endowed with a particular, coherent interpretation. This is, of course, a domain-independent matter. In this section, we consider the essential nature of nodes and links in object-centered knowledge representation schemes. We survey the various ways of looking at nodes and then examine the corresponding perspectives on links, paying particular attention to the *is-a* links which form the backbone of all taxonomic systems. We observe that the representation content can be viewed as assertional or merely structural, and consider the implications of each. A number of papers have examined these foundational issues, including [Woods 75, Brachman 79, Brachman 83, Sowa 87]. When considering the myriad interpretations of the network formalism, recall Brachman's prescription for considering representations at different levels of abstraction, and note that the shortcomings of many object-centered formalisms can be traced to their casual confusion of these levels and their failure to achieve consistency at every level. Throughout this section, we confine ourselves to issues directly tied to the node and link paradigm.

3.1 Nodes

Nodes, by themselves, can be conceived of in numerous ways. Brachman mentions a partial list in [Brachman 83]. In each case we offer our own example to suggest their intent.

- **set:** The collection of all elephants.
- **concept:** The mental idea of an elephant.
- **kind:** The qualities which are sufficient to make an object an elephant.
- **predicate:** A condition which is true of any elephant.
- **proposition:** An assertion of being an elephant.
- **prototype:** A representative elephant.
- **description:** The attributes of an elephant.
- **depiction:** The appearance of an elephant.
- **general term:** That which stands for elephants.[11]
- **individual term:** That which stands for a particular elephant.
- **individual:** A particular elephant itself.

When designing or working with a knowledge representation formalism, it is essential to be clear and consistent about how nodes are to be understood.

[11] In predicate calculus, terms serve to name objects in the domain of discourse. Terms may be variables, object constants, or functional expressions.

3.1.1 Generic versus Individual Nodes

Some systems, including most in the KL-ONE family, support a distinction between generic and individual concepts. Intuitively, a generic **elephant** concept represents a certain class of large gray quadruped mammals with trunks while an individual concept which we may name **Clyde** represents a particular instance of that class. It should be apparent that individuals are necessarily leaf nodes in the taxonomy.[12] Alas, a rigorous distinction between generics and individuals is a deep and long-standing problem in philosophy. Although the generic/individual dichotomy seems intuitively appealing, to date, no one has identified truly useful inferences which result from it.

3.1.2 Intension versus Extension

The *intension* of a concept is its intrinsic meaning. An intensional definition of a concept is made in terms of its attributes, or perhaps criteria for individuals denoted by that concept. The *extension* of a concept is the set it denotes. An extensional concept definition is made by listing the individuals it refers to. In Frege's classic example, the terms *morning star* and *evening star* have the same extension, i.e., the planet Venus, but entirely different intensions. Woods argues convincingly that a knowledge representation should support intensional descriptions independent of any possible extension [Woods 75]. Consider that unicorns, though mythological creatures, can be described and reasoned about in detail. Furthermore, distinct intensional descriptions should not imply a commitment to distinct extensions since objects may change over time. Hays' networks were the first to distinguish between *manifestations* of an entity [Hays 73]. All the definitional taxonomy systems we survey in this paper predominantly define generic concepts intensionally and organize their taxonomies according to intensional definition. In some cases, by adopting a closed world assumption[13] [Reiter 78], one can obtain an extensional definition of a generic concept by collecting the individual concepts located underneath it in the taxonomy.

3.2 Links

Links (or slots in frame terminology) can be construed as attribute/value pairs for the node where they originate. In the case of nodes representing generic concepts, a system such as KEE provides separate notational mechanisms for expressing facts about members of a class (member slots) and facts about the class itself (own slots) [KEE User's Guide 88]. For example, the elephant class night have a member slot for age and an own slot for the oldest elephant.

The attribute/value characterization of links seems clear enough when Clyde's age link points to 17, since 17 directly symbolizes the value of the attribute. When Clyde's age is known only to be greater than 16, however, the attribution becomes a predicate which must be true of the (uncertain) value. Of course, attributes alone are inadequate to represent many natural language constructs. This suggests that links need not be limited to expressing attributes of nodes. More generally, they may convey arbitrary relations between nodes. Since all attributes

---

[12] Ignoring distinct manifestations of an individual. For example, a baby changes into an adult as the aging process unfolds, yet its identity remains the same.

[13] Assume that any conclusion which is not derivable from the knowledge base is false.

of a node can be recast as predicates which must be true of the node, we can easily obtain a unified general interpretation in which all links point to a predicate which must hold for the node.

Links represent binary relations directly. To represent a relation of arity greater than two, one possibility is to decompose it into binary relations. We would therefore introduce a *node* to model a semantic relation, with the binary components emanating from it. Contrast this with earlier examples where we delegated the task of modeling semantic relations to links. This new technique engenders an entirely different network structure. A concrete methodology is suggested by Fillmore's *cases*, which can be characterized simplistically as nominals in semantic relation to a verb [Fillmore 68]. A node representing a verb has a set of links representing its cases. In Figure 3-1 the bitransitive verb ''to feed'' is represented in general terms as an action involving two animate entities plus food. Also shown is a particular instance of feeding in which Clyde feeds peanuts to Jumbo.

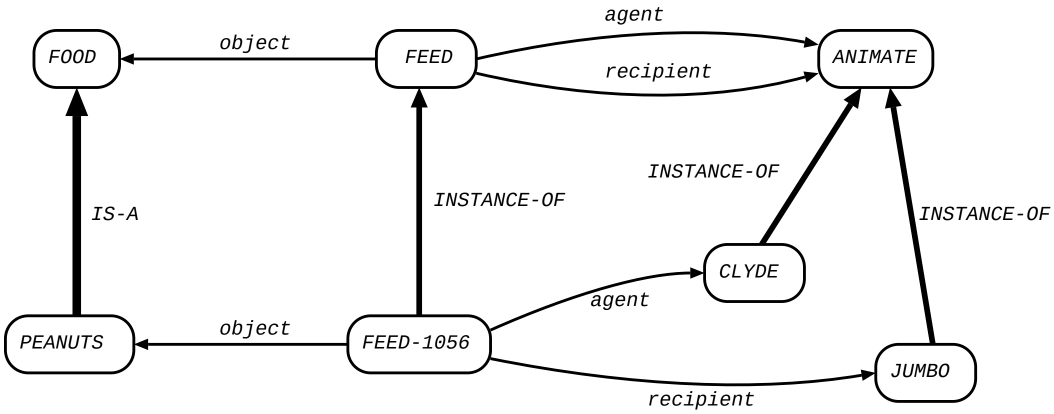

Figure 3-1: Case Decomposition of *Clyde feeds peanuts to Jumbo*

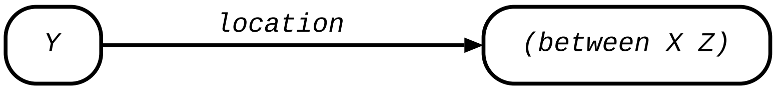

Figure 3-2: Y is between X and Z

Instead of decomposing a relation with three or more arguments, one could have a binary link that points to a parameterized construct. Borrowing an example from [Woods 75], ''**y** is between **x** and **z**'' might be represented as shown in Figure 3-2. The object at the end of the link is understood as a lambda expression applicable to anything which points to it, e.g.,

```
(lambda (u) (between x u z))
```

In retrospect, it seems that the emphasis on binary links in object-centered knowledge representations stems, in part, from their historical association with graphical notation. Recently, Schmolze has proposed a fairly natural extension to n-ary relations for systems based on a definitional taxonomy [Schmolze 89a].

Although we have implicitly treated links as declarative entities to this point, executable procedures may sometimes be associated with links as well. This is known as *procedural attachment*. Possibilities include procedures to be run when a link is added, deleted, queried or modified. Attached procedures, also known as *demons*, can be written in a domain-specific manner and can make reference to the current problem solving context. They can provide an interface to an encompassing system. Indeed, some AI problem solvers are structured as a collection of attached procedures and the reasoning process is driven by their interaction. Procedures allow *ad hoc* solutions to problems that are not yet well understood. Since attached procedures are unconstrained, they have the potential to undermine the tractability of a knowledge representation service.

3.3 The Ubiquitous IS-A

The systems we survey later in this paper all provide an explicit taxonomic hierarchy of nodes built with taxonomic links between pairs of nodes. Knowledge representation researchers have used many different labels for the taxonomic link reflecting their different views of taxonomy. These names include is-a, a kind of (AKO), superset and superc[14]. Sowa discusses some issues highlighted by such names in [Sowa 87]. Whereas all links in early semantic networks were accorded equal status, it is now recognized that taxonomic links are of paramount structural importance and it has become standard practice to distinguish taxonomic links for the purposes of examining and processing the network. We have already introduced the useful feature of *property inheritance* in which information is explicitly attributed to the most general concept where it applies and implicitly attributed to each of that concept's descendants. Thus, information flows downwards in the taxonomy along taxonomic links. Many of the issues surrounding taxonomic links have to do with inheritance. Taxonomic issues pertaining to inheritance are treated later in section 4 where we see that inheritance, despite its practical utility, is strictly a matter of implementation. In this section we are concerned with the intrinsic meaning of the taxonomic relationship.

3.3.1 How are IS-A links used?

Recall that many systems which distinguish between nodes representing generic and individual concepts also provide two distinct taxonomic links: one for the relation from individual to generic and another from a more specific generic to a more general generic. In KL-ONE, there might be an *individuates* relationship from the individual **Clyde** to the generic concept **elephant** and a *superc* relationship from **elephant** to **mammal**. Of course, Clyde inherits information from both elephant and mammal.

The most thorough discussion of taxonomic links is found in [Brachman 83]. Brachman identifies four categories of taxonomic relations between generic concepts. Notice how the categories overlap. We feel that the differences are a matter of nuance rather than sharp contrast. The categories are:

---

[14]For super concept.

1. **Subset/superset:** The set containment relationship is suitable for taxonomizing nodes which denote sets. For instance, the fact that a node denoting elephants subsumes another node denoting Asian elephants may be taken to mean that each element in the set of all Asian elephants is a member of the extension of the set of elephants.

2. **Generalization/specialization:** This form of relationship expresses a universally quantified conditional where the nodes are interpreted as predicates. In a straightforward analogue of our previous example, *asian-elephant(x)* is a one-place predicate, *elephant(x)* is a more general one-place predicate and the taxonomic relationship between the two is taken as $\forall x\, asian\text{–}elephant(x) \rightarrow elephant(x)$. If one admits exceptions, as in default logic, the situation is more complicated.

3. **AKO:** Brachman suggests that AKO is a restricted version of the generalization relationship which implies that the nodes involved are *kinds* rather than arbitrary predicates.

4. **Conceptual containment:** Here, the taxonomic relationship is meant to convey the notion that one description includes another in the sense of lambda abstraction, e.g., ‘‘To be a triangle is to be a polygon with three sides.’’

Brachman also gives several interpretations which may be placed on the taxonomic relationship from an individual to a generic. Naturally there are some strong parallels to the generic/generic case.

1. **Set membership:** An individual concept such as Clyde may be taken as an element of the set denoted by a generic concept such as elephants.

2. **Predication:** When a generic is construed as a one place predicate, subsumption of an individual by that generic may be seen as application of the predicate to the individual in question, e.g., *elephant(Clyde)*.

3. **Conceptual containment:** When individuals are structured descriptions, as in KL-ONE, the generic may be used as a guide to construction of the individual.

4. **Abstraction:** One may abstract from a generic concept to a prototypical individual, i.e., a canonical representative of the set. This is exemplified by *the elephant* in ‘‘Poachers value the elephant for its ivory tusks’’.

Some investigators have suggested a general purpose, programmable taxonomic link that would account for various types of inheritance, including idiosyncratic cases, by allowing the user to pick and choose individual semantic subcomponents of inheritance. Fox, for example, proposed the primitives *pass*, *add*, *exclude* and *substitute* [Fox 79]. As Brachman remarks, the semantics of these inheritance relations cannot be predicted in general [Brachman 83].

3.4 Facets

It is worth stressing that links and nodes are usually not atomic entities, rather they include elements known as *facets*. Facets attribute useful information, including meta-knowledge, to the constituents of a knowledge representation. For links, beside conveying the possible values that some attribute of a node may take on, facets may contain default values to be assumed in the absence of more specific information, upper and lower bounds on the number of values, procedures to be executed when the link is processed, and documentation, among other things.

Similarly, a facet on a generic node might indicate a prototypical instance of the concept it represents.

3.5 Relative Status Of Links And Nodes

In many semantic networks, links are fundamentally associated with the node from which they emanate. KL-ONE was the first to accord first class status to relationships. As we shall see in section 6.2 they are organized into their own taxonomy; that is, they can be specialized into more constrained subroles. Thus, all the advantages of taxonomizing nodes accrue to links.

3.6 Reference

Some applications, such as natural language processing, must deal with definite, indefinite and anaphoric references to individuals, e.g., *the (particular) elephant*, *an (unknown) elephant* and *the (previously mentioned) elephant*. Moreover, these references can be factual or hypothetical. Schubert's system was the first semantic network to handle definite and indefinite references [Schubert 76]. For example, his system represents the definite reference ''*John's car* is red'' as the graphical equivalent of:

$$\exists x\ \forall y[owns(John,y)\ \&\ car(y)\ \equiv\ x{=}y]\ \&\ red(x)$$

and the indefinite reference ''John owns *a red car*'' as the graphical equivalent of:

$$\exists x[owns(John,x)\ \&\ car(x)\ \&\ red(x)]$$

3.7 Assertional Force

In many representations, the mere presence of nodes and links carries assertional force. Woods argued that for the purpose of storing human verbal knowledge, a semantic network must be able to represent propositions independent of of their truth value or the system's belief in their truth value [Woods 75]. Others have recognized advantages in separating the definitional and assertional functions. Indeed, most of the systems we examine in this paper support these functions with separate components: a terminological component provides a vocabulary for making statements about the world and an assertional component states what is actually true in the real world. Additional complications arise when attempting to reason about a variety of possible worlds, but we do not consider this issue.

3.8 Conclusion

This section has examined the semantics of nodes and links. For nodes, we considered numerous ways of interpreting them, compared generics and individuals, and contrasted intension with extension. For links in general, we drew distinctions between (1) attributes of all members of a class and the class itself, (2) attributes as values and as predicates, (3) binary and n-ary relationships, and (4) declarative and procedural specification. For distinguished *is-a* links in particular, we reviewed Brachman's categorization. More generally, we touched upon the matters of granting links equal status with nodes, handling direct, indirect and anaphoric reference to individuals, and definition versus assertion. All of these questions are still controversial and the knowledge representation community has yet to settle on definitive answers. Rather than taking their own stand on each issue, Greiner and Lenat observed that ''The field of AI is strewn with knowledge representation languages'', many designed for a particular application domain, and they sought to achieve a general solution with a *representation language language* named RLL [Greiner and Lenat 80]. It allowed the user to control all aspects of a frame-like representation facility, including a selection of some 106 (!) types of slots, procedural attachment and inheritance. Of course, RLL passed all of the difficult questions along to the user, and it did not gain a wide audience.

4. Taxonomy and Inheritance

A taxonomy abstracts conceptual knowledge into a hierarchy of concepts where abstract concepts capture the commonality among their subordinates. While the features of an abstract concept could be explicitly attributed to each of its subordinates, that is unnecessary. Rather, the commonality of an abstraction may be implicitly attributed to its subordinates by virtue of their position underneath the abstract concept in the taxonomy. This attribution is called *inheritance*; subordinates are said to *inherit* the characteristics of abstract concepts.

4.1 Inheritance

Although inheritance does not extend expressive power, it does offer two intertwined advantages with important consequences: economy of description and localization of distinguishing information. Inheritance is notationally efficient because attributes may be specified in one place, at the most general concept in the taxonomy to which they apply, and implicitly attributed to subordinate concepts. That is, a concept may be defined simply by naming the general concept it specializes and indicating any further restrictions. Storage may thereby be utilized parsimoniously. Redundancy is minimized, easing update operations and promoting consistency. Furthermore, localization of distinguishing features speeds searching for concepts that match given features.

4.2 Defaults and Prototypes

For each attribute that characterizes a particular generic concept, we could specify a *default* value which the system will attribute to all specializations of that concept via inheritance, in the absence of more specific information or explicit information to the contrary. Recall, for example, that if the default color of elephants is grey, Clyde is an elephant, and Clyde's color attribute is unspecified, we may assume that Clyde is grey.

It is possible to form a *prototype* which describes a stereotypical specialization of some generic concept by grouping together a set of default values for its attributes. Prototypes are useful as typical examples and as default assumptions. For instance, we could define a prototypical CS faculty member at Columbia to be 35 years old, untenured and interested in software. We could also choose to define prototypes for more specific categories such as software faculty. This does not imply that any individual faculty member matches the prototype exactly. Indeed, an individual's attributes may conflict with any or all of the prototype's attributes.

4.3 Exceptions and Cancellation of Inheritance

Clearly it is useful to make generalizations, and inheritance is a good way to take computational advantage of them. Mandatory inheritance, however, is unrealistic because in reality there are many exceptions to generalizations. The canonical example runs as follows:

- Birds fly.
- Ostriches are birds.
- Ostriches do not fly.

Thus, Ostriches are an exception to the generalization that birds fly. We would like a system which allows us to retain the generalization despite the exception. At the same time, we would like to build a taxonomy in which ostriches inherit most of those characteristics from birds. We would like to *cancel*, or override, the ‘‘can fly’’ attribute for ostriches. The fact is that very few instances of concepts are exactly like the canonical case. This suggests pervasive use of cancellation to represent real world knowledge in concert with inheritance. On the other hand, cancellation presents problems of its own, as we shall now see.

4.4 Definitions versus Defaults

Much of the impetus for inheritance of defaults stems from Minsky’s influential paper on frames [Minsky 75]. The notion of cancelling inheritance seems perfectly natural, but perhaps it could also be described as seductive. Cancellation allows any concept to be placed anywhere in the taxonomy, and requires search to be exhaustive. Consider placing an arbitrary concept **X** underneath any other concept **Y** in the taxonomy. With cancellation this can always be done. Simply specify X by cancelling all properties of Y which are not true of X and adding all properties of X which Y does not exhibit. This point is neatly illustrated by a delightful anecdotal example due to Brachman, consisting of a question and a series of answers [Brachman 85]:

Q: What’s big and gray, has a trunk, and lives in the trees?

A1: An elephant, I lied about the trees.

A2: A giraffe -- I lied about the color, the trunk and the trees.

A3: An idea -- I lied about the color, the trunk, the trees, and about the ‘‘lives’’.

The problem, however, is a serious one. When cancellation is permitted in a taxonomy, the taxonomy is reduced from providing definitions to providing defaults.[15] This is not to say that cancellation of inheritance is necessarily bad. Rather, the consequences of compromising the taxonomy’s informational and organizational force in return for flexibility should be carefully considered.

Systems like KL-ONE insist on an *enforced semantics* wherein cancellation of inheritance is forbidden. One concept subsumes another concept if and only if every individual described by the second is also described by the first. Subsumption is reflexive, anti-symmetric and transitive. As a result, each concept is properly subsumed by its ancestors. Equivalently, the subsumption relation imposes a partial ordering on the taxonomy. This speeds searching and matching in the taxonomy by fixing the location of a concept based on its definition. For example, with enforced semantics, one can efficiently determine if a concept matches some generic concept: just look at its ancestors in the taxonomy. It is also efficient to find all the instances of a generic concept by searching among its descendants. Perhaps more importantly, the knowledge base designer is forced to make careful decisions in specifying concepts. Bear in mind, though, that KL-ONE’s ability to capture important generalizations is hindered by its inability to cancel inheritance.

[15]Recalling the example from section 1.5.3, notice that a generic concept which defines a restricted value for a particular attribute can also provide a default value for that attribute. There is no contradiction unless the default value contradicts the value restriction.

4.5 Multiple Inheritance

A concept may have more than one parent in the taxonomy.[16] That often reflects a natural state of affairs. For example, a professor is both a teacher and a researcher. Given **teacher** and **researcher** concepts, we would like a **professor** concept to inherit information from each of them. This is called multiple inheritance. Suppose different parent concepts have different values for the same attribute. If the values are consistent we can just take their logical conjunction. This always works in the case of definitional taxonomies. But there is a problem for systems where the values can be inconsistent: we need a method for resolving the inconsistency. The inheritance algorithm should make plausible inferences about the attributes of a concept obtained via inheritance. One possibility would be to perform a breadth-first search through a concept's ancestors in order to select the value for a particular attribute. Another possibility would be to perform a depth-first search. Some systems offer a variety of search strategies. Some even allow users to program their own strategies. Each of these alternatives has an *ad hoc* flavor. As Touretzky illustrates, the combination of multiple inheritance with exceptions gives rise to potential contradictions [Touretzky 84, Touretzky 86]. One contribution of his thesis was the *inferential distance ordering* which indicates exactly when inheritance is ambiguous, and exactly how to inherit information when it is unambiguous. Consider the following well-known example:

- Quakers are Pacifists.
- Republicans are not pacifists.
- Nixon is a Quaker.
- Nixon is a Republican.

Is Nixon a pacifist or not? Or is the answer ambiguous? Essentially, Touretzky says that we may prefer one answer to another if and only if a single inference path allows us to determine that the preferred answer is more specific. In his words [Touretzky 86]:

> Suppose we are trying to find out if A has property P. If A inherits from B which has property P, and also from C which has property ~P, what conclusion should we reach about A? The inferential distance ordering says: if A has an inference path via B to C and not *vice versa*, then conclude that A has P; if A has an inference path via C to B and not *vice versa*, then conclude that A has ~P; otherwise there is an ambiguity.

Among his other contributions, Touretzky showed how redundant information should be handled correctly in the face of multiple inheritance.

4.6 Conclusion

We have noted that the advantages of taxonomies with inheritance include (1) economy of expression, (2) localization of distinguishing information, (3) parsimonious use of storage, and (4) speedier search. Defaults and prototypes can further increase the utility of taxonomic inheritance. Exceptions to inheritance seem natural and foster flexibility, so they have seen widespread use in KR systems ranging from early research efforts such as KRL and FRL through present day commercial systems such as KEE and ART. On the other hand, the use of exceptions reduces taxonomies from making definitions to providing defaults and, arguably,

[16] So the taxonomy forms a directed acyclic graph, sometimes called a tangled hierarchy, and perhaps a lattice.

encourages sloppy knowledge engineering. Moreover, cancellation undermines a taxonomy's ability to guide search. For these reasons, research systems such as KL-ONE and its successors have opted for definitional taxonomies where exceptions are precluded. Multiple inheritance in the face of conflicting information is also a thorny problem for systems with non-definitional taxonomies, and it is presently an active area of inquiry. Systems with definitional taxonomies admit a special inference known as classification which is the subject of the next section. We will conclude our discussion of classification with an assessment of the profound impact definitional taxonomies and classification have on those KR systems in which they are adopted.

5. Automatic Classification

This section is devoted to automatic placement, or classification, of structured concepts in a definitional taxonomy. We shall consider classification of a newly defined concept, reclassification of a redefined concept and incremental classification of a concept as its definition is being completed. We then summarize some tractability results for computing subsumption and finally discuss some assumptions about the proper nature and role of classification that characterize all of the existing systems based on classification.

5.1 Classification

Classification is a process which places concepts into a taxonomy according to their definition by computing subsumption relationships. The result is that the classifier establishes the correct taxonomic links among concepts. The classifier employs a special purpose algorithm which adds concepts to the taxonomy one at a time, taking advantage of the partially completed taxonomy's hierarchical organization. The process of classifying an individual concept, i.e., determining the most specific set of concepts which describe it has also been called *realization* [Mark 81]. Automatic classification is useful for incremental construction of a taxonomy, enforcing semantics, type checking, and pattern matching.

Subsumption can be computed efficiently by a special purpose algorithm. For terminological representations, concept attributes include roles and relationships among them, e.g., NIKL's role constraints (refer to section 6.2). One concept subsumes another if and only if each attribute of the first (recursively) subsumes some attribute of the second. Thus, every role of the first concept must subsume a role of the second, which is in turn only true if the value restriction and number restrictions of the first role subsume their counterparts in the second role. This criterion assumes that concept definitions specify both necessary and sufficient conditions. When only necessary conditions can be specified, a concept is deliberately distinguished as a primitive concept. Since a primitive concept lacks sufficient conditions, the classifier has no basis for automatically placing other concepts underneath it. Nonetheless, the user can explicitly define a concept to be subsumed by a certain primitive or primitives. It need not be primitive itself. For example, even though the concepts for **person** and **female** may be primitive, the concept for **woman** is fully defined as their logical conjunction. As an extension to definitional classification, the K-Rep system supports heuristic rules that augment conditions used to determine class membership.

How should a concept be installed in a taxonomy? From the definition of subsumption, there is exactly one place in the taxonomy where a concept belongs. It should be located underneath its most specific subsumers and simultaneously above its most general subsumees. Classification easily detects the definition of a redundant concept since it would coincide with an existing concept. Likewise, classification detects a logically impossible, or *incoherent*, concept when it cannot find a suitable place for the concept in the taxonomy. Rather than permitting this, the concepts can be collapsed together.

The classification process can be automated with reasonable efficiency. Schmolze and Lipkis formally specify the classification algorithm in KL-ONE [Schmolze and Lipkis 83]. NIKL's classifier is described in [Robins 86] and LOOM's is presented in [MacGregor 88].

Circular concept definitions are conceivable because concept definitions reference other concepts by name. If concept **A** is defined to be a subconcept of **B** then **A**'s definition is said to *depend* on concept **B**. Also, **A** depends on **C** if **A** has a role whose value restriction is **C**.[17]A sequence of concept dependencies may introduce a cycle. As a trivial example, one might define the spouse of a human to be a human. It is difficult to implement a classifier which processes all circular definitions correctly; indeed the problem is not entirely understood. Some details of NIKL's technique for classification with respect to cycles are in [Kaczmarek *et al.* 86]. Although the KL-ONE and NIKL languages permitted cycles and their classifiers were designed to handle circular definitions, the results were not satisfactory. Certain troublesome examples are known, and no one has been able to identify precisely when classification with cycles is correct and when it is not. Later systems have simply forbidden cycles. Of course, this limits the expressive power of the language. Recent theoretical work examining the semantics and computational properties of terminological cycles and is found in [Baader 90] and [Nebel 90a].

Classification supports type checking. When a concept is defined to specialize some other concept, the classifier will verify the subsumption relationship between the two. If the verification fails, a type error is indicated.

Classification supports pattern matching against the knowledge base. Given a new concept which specifies a desired pattern, the classifier places it into the taxonomy. All concepts which match the pattern will then be subsumed by the pattern concept. Further search may then be conducted on the matched concepts, as appropriate. The ARGON information retrieval system relies on the classification facility of KANDOR [Patel-Schneider *et. al.* 84], discussed in section 6.3. ARGON drew heavily from the design of an earlier system, RABBIT [Tou *et al.* 82]. The LaSSIE system based on ARGON mantains information about a large software system [Devanbu *et al.* 89]. Another classification-based query processing facility is CANDIDE [Beck *et al.* 89].

5.2 Reclassification

If concepts are modified, either during problem solving or during interactive editing of the knowledge base, those concepts must be reclassified in order to maintain the integrity of the definitional taxonomy. Current systems, e.g., the KREME interface to NIKL [Abrett and Burstein 87, Abrett *et al.* 87] LOOM and K-Rep support interactive, incremental modification of concepts. One major concern is that the reclassification process be swift, or at least that lengthy reclassification operations be identified heuristically during interactive editing.

It is possible to sharply circumscribe the set of concepts that a modified concept must be compared with during reclassification by recursively examining its taxonomic neighbors to determine whether, and how far, it can move up or down in the hierarchy. Interestingly, a concept can move both up and down after modification [Balzac 86]. Broadly speaking, addition of attributes may cause a concept to move downward, deletions may cause it to move upward, and modifications may cause movement upward, downward, or both. Not all changes result in movement, however.

It is important to notice that reclassification of a concept may in turn require reclassification of any or all concepts whose definitions depend on it. Reclassification of dependent concepts is

[17]Dependencies may be introduced in other ways as well, e.g., in a chain of a role constraint.

a recursive process. The nature of a modification can limit the set of dependent concepts which must be considered during reclassification. Consider a concept whose value restriction has been modified. It is not necessary to reclassify all of its subsumees, just those which inherit that value restriction. In the worst (pathological) case, reclassification of a single concept necessitates reclassification of every concept in the knowledge base.

5.3 Interactive Classification

Finin and Silverman built a prototype interactive classifier to explore how an existing knowledge base might be used to assist in interactive knowledge acquisition [Finin and Silverman 86]. In particular, their system could be used to extend a KL-ONE style knowledge base by adding one new node (a role or a concept) to the taxonomy at a time. Since these additions can be made generalizations of existing nodes, knowledge about existing nodes can be added implicitly. With standard classifiers, a user is expected to define a node completely, whereupon the classifier places it in the taxonomy. If the resulting location suggests that the node does not relate to other nodes as expected, the user must modify the definition and the process is repeated, perhaps several times. The advantage of interaction lies in the opportunity for a knowledge engineer to refine a definition and step through the classification process on an incremental basis.

First the user provides an initial description of the new node by stating some of its attributes or subsumers. The next step is to ascertain the most specific subsumer of the new node, a process which naturally proceeds downward from the root of the taxonomy. Two strategies are used to expedite the MSS search: classification by *exclusion* and classification by *attribute profile*. The task is simplified by an important limitation of Finin and Silverman's system: they restrict the hierarchy to tree form. The final task, finding the most general subsumees of the new node, is greatly simplified by the same restriction.

Classification by *exclusion* takes advantage of the tree structure in that there is always one existing node, called the *current MSS*, which is the most specific subsumer of the new node that has been found so far. If none of the children of the current MSS are consistent with the description of the new node, then the current MSS is the final MSS. Otherwise, since the taxonomy is tree structure, only one child of the current MSS can become the next current MSS. If the system can eliminate all but one child based on its knowledge of the new node, the user is asked to confirm, deny or restrict the attributes of that child as applicable to the new node. These responses determine either a final MSS (the parent) or a new current MSS (the child). When more than one child of the current MSS is consistent with the new node, the system tries to exclude some of them as quickly as possible by heuristically choosing an attribute and asking the user about it *vis a vis* the new concept.

The attribute profile method seeks to speed the search process heuristically. It proceeds by choosing an attribute from the new node's initial description, locating the most specific node in the taxonomy which subsumes all existing nodes having that attribute, and proposing that node to the user as a subsumer of the new concept. If the user concurs, the proposed node becomes the current MSS and the process continues from that location until a final MSS is arrived at.

Finin and Silverman's interactive classifier is limited in several ways, principally in the restrictions that nodes can only have one parent, and that concepts and roles can not be deleted or modified. Their paper proposes extensions to eliminate these constraints and otherwise increase the system's usefulness, however extending the classifier to a complete KL-ONE-like language proved difficult [Kass *et al.* 88]. With a tree-structured taxonomy, any pair of concepts are either in a subsumption relationship to one another or they are disjoint. With multiple inheritance, that is not so and interactive classification of a new concept must compare it with most of the existing concepts.

5.4 Tractability of Computing Subsumption

A seemingly insignificant extension to the expressiveness of a representation language may drastically compromise its tractability. Brachman and Levesque focus their analysis on one such ‘‘crossover point’’ in the computation of subsumption relationships in frame description languages [Brachman and Levesque 84][18]. They examine a typical language for which subsumption can be computed in $O(n^2)$ time. Next, they show that an apparently simple variant of that language is co-NP-complete. This leads to the conclusion that one must make careful choices in trading expressiveness for tractability. Moreover, there is no single best choice. Instead, different choices may complement one another nicely. This is the motivation for the hybrid representation systems we survey in section 7.

The result of Brachman and Levesque has practical significance since their co-NP-complete language is a subset of the terminological languages employed by such systems as KL-ONE, NIKL and KL-TWO. Nebel later showed that for another subset of the languages used in such systems as KL-ONE, NIKL, KL-TWO, KANDOR and BACK, subsumption is NP-hard [Nebel 88]. Patel-Schneider demonstrated that subsumption in NIKL and similar systems is undecidable, as well [Patel-Schneider 89]. By showing that no complete algorithm for such languages is possible, his result underscored the trend towards consciously incomplete subsumption and classification algorithms that was instigated by the intractability of subsumption in languages like NIKL. Recently, Nebel showed that subsumption in terminologies is inherently intractable (co-NP-complete) [Nebel 90b].

5.5 How Definitional Taxonomy and Classification Influence Knowledge Representation

Automatic classification is useful for constructing a definitional taxonomy and maintaining its consistency, but classification is computationally burdensome. Levesque and Brachman have argued that a knowledge representation facility should be correct (sound and complete), yet dependably quick enough for the most critical applications [Levesque and Brachman 87]. Since classification is the principal and most expensive operation provided by the systems we survey in this paper, it would follow that such systems should restrict their languages so that tractable (polynomial time) and guaranteed correct classification operations can be assured in the worst case. Doyle and Patil have called this conclusion the *restricted language thesis* [Doyle and Patil 89]. They counter with several strong arguments, based on their experience using NIKL in medical expert systems:

- Pursuing the restricted language thesis ‘‘destroys the generality of the language and

---

[18] Much of which is included in [Levesque and Brachman 85].

the system.''

- Desired asymptotic efficiency can be achieved without omitting useful but problematic constructs entirely, rather they should be employed judiciously.

- In most domains, the proportion of ''natural kind'' primitive concepts is substantial. Severe language restrictions make it impossible to define many other concepts, which must therefore be declared primitive as well. Since primitive concepts cannot be classified, the utility of classification is diminished by these ''fake'' primitives.

- The emphasis on runtime efficiency is misguided, because general purpose systems should not focus on worst-case performance for the most critical applications. Rather, (1) the general case is also important, (2) other costs, such as space, should be considered, and (3) other inferences besides classification should be taken into account.

In any case, Nebel's recent result rules out the possibility of tractable worst-case performance for any reasonably useful classification-based system.

We have already mentioned the claims that knowledge representation systems can achieve improved clarity and enhanced performance with distinct, separately optimized definitional and assertional components. This suggests to some that classification should operate exclusively on terminological definitions, a judgement that Doyle and Patil named the *restricted classification thesis*. They argued that this restriction significantly reduces the value of classification in practical applications where useful classifications should involve contingent information. They cite the classification of treatable and non-treatable diseases, which depends on the evolution of medical knowledge.

We share the conviction of Doyle and Patil that definitional classification is useful as a special purpose inference to be employed under carefully controlled circumstances, and that it is inadequate as a foundation for general purpose knowledge representation. In the future, it may be productive to distinguish the use of classification for knowledge acquisition from its role in run-time inferencing.

6. Definitional Taxonomy Systems

Having established a framework for studying object-centered formalisms, we devote this section to a survey of existing knowledge representation systems that share three distinguishing characteristics:

1. They employ definitional taxonomies with enforced semantics and therefore support automatic classification of concepts in the taxonomy.
2. They restrict the expressive power of their language to achieve relatively efficient performance.
3. They explicitly distinguish between terminological and assertional information.

Such systems have been coupled with production systems or theorem provers to form so-called hybrid systems which are discussed in section 7.

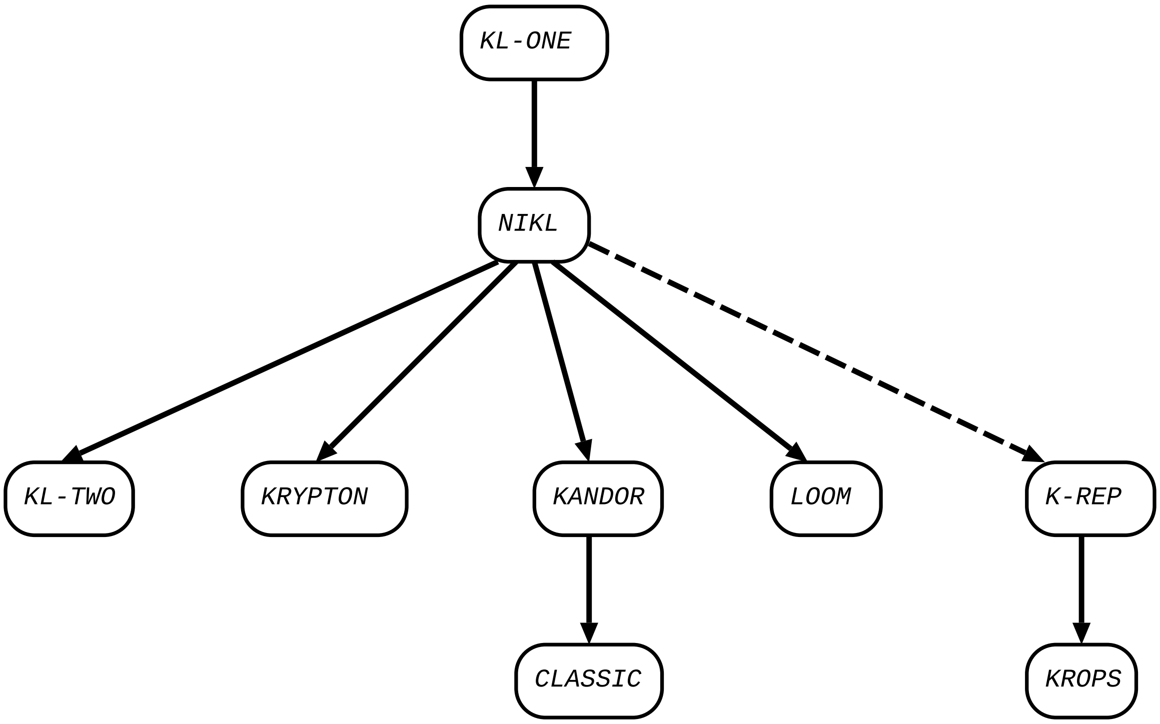

Figure 6-1: Family Tree

6.1 Genealogy

We now sketch the ancestry of definitional taxonomy systems and related hybrid systems. A rough family tree is shown in figure 6-1. The seminal work in this area was KL-ONE, begun at Bolt, Beranek and Newman in the late nineteen-seventies. NIKL[19] was an improved version of KL-ONE [Moser 83, Robins 86, Kaczmarek *et al.* 86, Schmolze 89b]. Development of NIKL continued until recently at USC/Information Sciences Institute (ISI). Two hybrid reasoning

[19] New Implementation of KL-ONE.

systems arose from this work: KL-TWO directly employed NIKL as a component [Vilain 85], while a new taxonomic reasoner was incorporated in KRYPTON [Brachman *et al.* 83a, Brachman *et al.* 83b, Brachman *et al.* 85]. Although the design of KL-ONE and NIKL was more carefully constrained than for earlier KR systems, it was soon discovered that classification in NIKL was neither computationally tractable in the worst case, nor was it complete. This lead to KANDOR, whose expressive power and inferential ability were specifically intended to achieve tractability [Patel-Schneider 84]. Computation of subsumption in KANDOR was later found to be NP-hard [Nebel 88]. A very recent follow-on to KANDOR which reflects current thinking is CLASSIC, now under way at AT&T Bell Laboratories [Borgida *et. al.* 89]. In the late eighties, ISI suspended work on NIKL and turned to a successor which they named LOOM [MacGregor and Bates 87]. A more recent conception of LOOM transforms it into a very ambitious, general purpose KR tool, however this effort is still in progress [MacGregor and Brill 89]. IBM's K-Rep system has much in common with the KL-ONE family, but is geared towards support of large-scale expert systems [Mays *et al.* 91a]. KROPS is a hybrid representation system which includes K-Rep [Daly *et al.* 88].

6.2 KL-ONE and NIKL: A Case Study

KL-ONE has proven to be a particularly influential system which raised important issues [Brachman and Schmolze 85]. We examine KL-ONE and NIKL as a single entity, and as a case study for introducing ideas common to other systems covered in this section. KL-ONE/NIKL ideas and terminology have evolved over time. We adopt the vocabulary of a very recent presentation [Schmolze 89b]. Both KL-ONE and NIKL were formally specified. Schmolze and Israel give a semantics for KL-ONE in first-order logic [Schmolze and Israel 83] and Schmolze defines NIKL in [Schmolze 89b].

NIKL supports a conceptual taxonomy composed of *generic* concepts and *individual* concepts.[20]Generic concepts define classes of entities whereas individual concepts define unique entities that hold membership in at least one generic class.[21] Generic concepts are *defined* if their specification provides both necessary and sufficient conditions for class membership; otherwise they are *primitive*. Primitive concepts are understood to entail certain sufficient conditions for class membership which are not or can not be expressed in NIKL notation. It is nevertheless expected that a broad set of necessary conditions can generally be given for primitive concepts. Concepts denoting *natural kinds*, including most common nouns such as chair, elephant and female, can only be partially defined in terms of their attributes and therefore must be primitive. All definitions must bottom out somewhere, so all defined concepts are ultimately expressed in terms of primitive concepts.

In a NIKL taxonomy, the *superc* link from one generic concept to another expresses class subsumption. A generic concept may have one or more *supercs*, or, superconcepts. Taken together, the superc links in a taxonomy define a partial order over potential class instances. An individual concept has one or more *individuates* links to generic concepts, each link denoting

[20]Or *structured conceptual objects*.

[21]The interpretation of individuals in NIKL has varied over the years, and even the designers don't agree completely.

class membership. The taxonomy is rooted at a maximally general concept, **thing**.

All concepts in the taxonomy inherit attributes from their subsumers. Each concept must be unique, i.e., it is defined as the conjunction of its subsumers plus at least one distinguishing characteristic. Indeed, if two concepts are given equivalent definitions, NIKL collapses them into one.[22] Since a concept's definition must implicitly conjoin the definitions of its parents, there can be no inconsistency arising from multiple inheritance.

In NIKL, *roles* express a *potential* binary relationship from one generic concept (the domain) to another concept (the range). For example, both the domain and range of the **parent** role would be **animal**. Roles are first class entities in NIKL, and they form their own taxonomy similar to the concept taxonomy. A role is said to *differentiate* roles which subordinate it in the taxonomy. For example, the **mother** and **father** roles would differentiate the **parent** role. The role taxonomy is rooted at a single, maximally general role known as **mostgeneralrole**, whose domain and range are **thing**. Roles are placed in their taxonomy by the classifier. Like concepts, they can be marked as primitive. In addition, a role may be specified as the inverse of another role.

An attribute of a concept is specified by a restriction of a role for that concept, called a *role restriction*, which has several components. The *value restriction* is a concept which denotes the class of fillers (values) that can potentially satisfy the role, e.g., the mother of a person must be a person. Role restrictions also have *number restrictions* consisting of a minimum cardinality and a maximum cardinality which express the lower and upper bound, respectively on the cardinality of potential role fillers. For the mother of a person, these would both be one, as each individual person has exactly one (biological) mother. In general, an individual might have multiple fillers for a role, e.g., six friends, so the minimum cardinality must be at least zero while the maximum cardinality may be any positive integer, or nil, denoting infinity.

Role constraints express a relationship between fillers of two of a concept's roles. They consist of a *constraint type*: equal, subset or superset, and two *role chains* which provide the arguments to the constraint type. Role chains are a sequence of role names which denote the composition of role relations. Consider the role constraint illustrated in Figure 6-2. It enforces the condition that the **tenured-faculty** of a department must be a subset of its **faculty**. The constraint type, subset, is shown in a diamond attached to the concept **department**. The role chains, shown as sequences of directed arcs emanating from the diamond, trace paths to the concepts **tenured-faculty** and **faculty**. Like roles, role constraints are definitional components of concepts, and are inherited by all subordinate concepts.

Descriptions of NIKL include a far more general mechanism for constraining the relationship among roles of a concept called *structural descriptions*. Apparently they were not implemented, at least not in full generality.

NIKL offers several other facilities which, however, are not fully taken into account by the classifier. A *disjointness* declaration states that a given set of concepts are mutually disjoint.

---

[22]K-Rep, however, does allow equivalent concepts. This is useful for interactive concept definition and refinement.

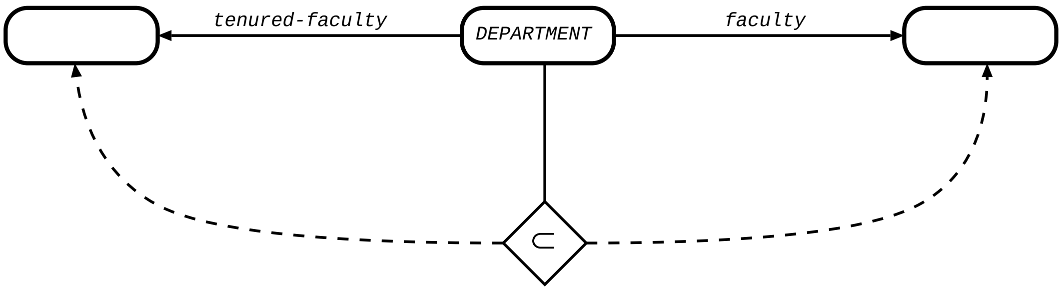

Figure 6-2: Simple Role Constraint

Similarly, NIKL has allowed *covering* declarations to indicate that a covered concept is exhausted by some set of concepts. The *attached data* facility allows a programmer to associate arbitrary information with a concept or role, and control whether or not it is inherited.

The KREME graphical editing environment was developed at Bolt, Beranek and Newman, Inc. for creating and browsing NIKL knowledge bases [Abrett and Burstein 87, Abrett *et al.* 87]. KREME also supports two limited types of automatic learning: it attempts to deduce useful generalizations of existing concepts and to learn by analogy from examples.

KL-ONE and NIKL constitute a landmark contribution to object-centered knowledge representation. As noted by Schmolze, the KL-ONE/NIKL effort produced several important contributions [Schmolze 89b]:

1. NIKL provides a separate role taxonomy, with subsumption and classification of roles.

2. NIKL can specify constraints among fillers of a concept's roles.

3. NIKL provides both primitive and defined terms.

4. NIKL has an automatic classifier, for both concepts and roles.

As well, their design represents the first explicit attempt to achieve consistency at the epistemological level of representation. NIKL has been criticized from opposite perspectives. Some of its designers became concerned that classification with the NIKL language is undecidable, and that NIKL's classification algorithm, though sound, is incomplete (section 5.4). This inspired them to pursue more restricted languages in search of tractable performance in the worst case. Conversely, Doyle and Patil argued that NIKL's language is already overly restrictive and that emphasis on worst-case tractability is misguided (section 5.5). We feel that the latter criticism is more significant. NIKL was nonetheless able to sustain a user community which produced a number of AI applications systems, especially in the area of natural language processing. To date, it has been the most widely used system of its kind.

6.3 KANDOR

KANDOR is a carefully circumscribed knowledge representation system that was designed according to the notion that ‘‘small is beautiful’’ [Patel-Schneider 84]. This philosophy is reflected in KANDOR's carefully defined and deliberately limited expressive power, inferential ability and interface. KANDOR was developed as part of ARGON, a system for quick interactive information retrieval from a large knowledge source [Patel-Schneider *et. al.* 84]. It has also been used in LaSSIE, a knowledge-based software information system, and with the TELI natural language interface [Devanbu *et al.* 89].

A KANDOR knowledge base is composed of two kinds of structured items: *individuals* are intended to model real-world entities and *frames* are intended to model collections thereof. Frames simply provide descriptions. They have no assertional import.

An individual has *slots* which map the individual into a set of *slot fillers* whose elements are individuals, strings or numbers. Frames describe a collection of individuals by means of conditions under which an individual belongs to the collection. Frames may be defined or primitive, just as in KL-ONE. The two kinds of conditions expressed by frames are *superframes* and *restrictions* on slot fillers. A frame's superframes are other frames which subsume it by definition. Frames can have three kinds of restrictions on slot fillers, as follows:

- A *some-restriction* states that some minimal number of the elements must be instances of a particular frame.
- A *max-restriction* places an upper limit on the number of elements.
- Under an *all-restriction*, every element must be an instance of a particular frame or be a particular individual.

These restrictions are sufficient to express a reasonable variety of constraints, including the sorts of cardinality constraints that could be expressed in KL-ONE. The following illustration patterned after [Patel-Schneider 84] shows a frame which describes certain graduate students. The first line invokes the **grad-student** superframe. The second line is a max-restriction which restricts the number of advisors to at most two. The some-restriction of the third line requires at least two (and therefore all) of the advisors to be professors.

DOUBLE-ADVISEE ::= [GRAD-STUDENT
advisor: $\leq$ 2
advisor: $\geq$ 2 PROFESSOR]

KANDOR does not have a taxonomy of slot-types similar to KL-ONE's role taxonomy, and so it is impossible to define one slot as a restriction of a different slot, e.g., academic-advisor as a specialization of a more general advisory relationship. KANDOR is also unable to express constraints among slots of a frame, as KL-ONE did with its role constraints.

KANDOR's representation was restricted in part by the requirement that computation of subsumption among concepts must be done efficiently and guaranteed to halt within a reasonable time. This restriction arises because KANDOR's interface must efficiently provide the set of individuals which match the description of a given frame. Suitable speed is attained using an explicit taxonomy which is built incrementally by classifying frames and realizing individuals as they are defined. KANDOR does not aspire to provide additional inferential capabilities beyond classification and realization.

Like KRYPTON (see section 7.3), KANDOR provides a strictly functional interface to a knowledge base so that it may be thought of as an abstract data type. The user can create individuals, slots and frames. She can also make certain queries regarding these items. This functional interface makes it possible for KANDOR to have a well-defined and implementation-independent semantics. Modifications are not allowed on the grounds that their semantics are not yet completely understood and also because of the computational expense. This seriously limits the practical value of KANDOR.

As an aside, KANDOR also includes a modest assertional component which, like a database, allows no disjunction, negation or quantification. Thus one can easily tell if an individual instantiates a particular frame. The limited power but high efficiency of KANDOR's assertional component stands in contrast to that of KRYPTON which uses full first order predicate calculus.

While KANDOR's expressiveness is more limited than KL-ONE's and it does not permit any updates, Patel-Schneider asserts that these drawbacks are more than compensated for by its strong points. He contends that KANDOR's small size, well-defined semantics and interface, and computationally tractable operations (due to its limited expressive and inferential power) are fundamental to its utility as part of ARGON. In sum, he holds that this carefully limited system is more usable than others which subscribe to the ‘‘big is beautiful’’ philosophy. We agree that KANDOR's design is especially neat and clean. On the other hand, though KANDOR took an extreme position with its restricted power, it nevertheless failed to meet one of its most important design goals because computation of subsumption in KANDOR is now known to be NP-hard [Nebel 88].

6.4 CLASSIC

CLASSIC is a follow-on to KANDOR and the latest member of the KL-ONE family now being produced by Brachman's group at AT&T Bell Laboratories [Borgida *et. al.* 89]. Like KANDOR, CLASSIC has been used with the ARGON, LaSSIE and TELI systems [Devanbu *et al.* 89]. The description in [Borgida *et. al.* 89] is notable because the authors make a concerted effort to present CLASSIC from a more traditional database perspective. An appealing feature of CLASSIC is the fact that it has a formal denotational semantics.

In CLASSIC, concepts are built with three kinds of complex constructors for describing their intensional structure:

- role value restrictions
- cardinality bounds
- co-reference constraints[23]

These constructors are illustrated in the following definition of the **grad-student** concept and its ensuing specialization:

[23] Equivalent to role value maps limited to an equality operator.

```
define-role[advisor];

define-role[department];

define-concept[GRAD-STUDENT,
              (AND (ALL advisor PROFESSOR)
                   (AT-LEAST 1 advisor))]

define-concept[GR-STD-WITH-ONE-ADVISOR-IN-DEPT,
         (AND GRAD-STUDENT
              (AT-MOST 1 advisor)
              (SAME-AS (department)
                       (advisor department)))]
```

Observe that roles have no internal structure in CLASSIC, like KANDOR but unlike earlier predecessors such as KL-ONE and NIKL. CLASSIC's concept definition language is compositional: expressions describing concepts may contain full concept expressions in many places. Expressions are composed via the *and* operator. Equivalent alternatives to a compositional expression may be used interchangeably.

CLASSIC provides a modest forward chaining facility which allows it to infer that if an individual matches one kind of concept, then that individual is also an instance of a second kind of concept. The following example, taken from [Borgida *et. al.* 89], states that any individual student belongs to the class of things that eat only junk food:

```
assert-rule[STUDENT, (ALL eat JUNK-FOOD)]
```

It would be undesirable to make (ALL eat JUNK-FOOD) part of the STUDENT concept definition, because then the classifier would only recognize[24] an individual as a student if it were known that she ate nothing but junk food. CLASSIC propagates the effects of rules until arriving at a fixed point.

CLASSIC has the usual notion of primitive concepts which support extensional grouping of individuals, plus *enumerated concepts* which directly name a time-invariant set of individuals, e.g.:

```
define-concept[tenured-ai-professor,
               (one-of Kender, McKeown, Stolfo)]
```

This mechanism undermines the separation of terminological information, i.e., concepts, from assertional information, i.e. the existence of individuals.

For the sake of integrity checking, the *disjoint primitive* constructor separates primitive subclasses of a common parent class, e.g.:

```
(disjoint-primitive student level graduate)
(disjoint-primitive student level undergrad)
```

In contrast with earlier members of the KL-ONE family, CLASSIC incorporates a procedural ''black box'' in the form of *test concepts*. Test concepts denote all entities such that

---

[24]I.e., automatically classify.

an associated predicate written in the host implementation language returns true, e.g., the following denotes even integers:

```
define-concept[even-integer,
               (AND integer
                    (test even)].
```

Test concepts specify primitive sufficient conditions. They are dual to the usual primitive marker which specifies primitive necessary conditions. Subsumption inference can, of course, become intractable if this facility is abused.

CLASSIC individuals are created by the *create-ind* operator and facts are attributed to them by the *assert-ind* operator, e.g.:

```
create-ind[Weida]
assert-ind[Weida,
           (AND GRAD-STUDENT
                (FILLS advisor McKeown)]
```

Here, **Weida** is an element of the class **grad-student**, and is related to another individual, **McKeown**, by the advisor role. Individuals need not be named; such requirement would be burdensome in the case of an extremely large knowledge base. For similar reasons, CLASSIC distinguishes *host* individuals from the usual individuals. These are primitive data values in the host implementation language[25] such as numbers or strings.

One can readily state incomplete information about individuals, for example, the following asserts that all members of the area paper committee for the individual named **Weida** have department **CS**.

```
assert-ind[Weida,
           (ALL area-committee-member
                (ALL department (ONE-OF cs)))]
```

Since CLASSIC aims to support incomplete information, it assumes open-world semantics by default.[26]The above assertions about Weida say nothing about whether he has other advisors, however one can explicitly assert that nothing else is related to an individual by a particular role, e.g.:

```
assert-ind[Weida, (CLOSE advisor)]
```

CLASSIC can also infer closure based on cardinality. Suppose that the **advisor** role of the **grad-student** concept is defined to have at most two fillers. Then, once a second advisor has been attributed to an individual student it is certain that no more are permitted.

CLASSIC supports various queries about concepts, role fillers and class membership, and a more powerful, integrated query language is in progress. Besides providing the set of existing individuals satisfying a given query, the facility would derive information necessarily true of all individuals which could satisfy a given query, independent of known instances, and taking into account any applicable rules.

---

[25]E.g., C or LISP.

[26]That is, by default it does not make a closed world assumption.

CLASSIC retains KANDOR's emphasis on a ''lean and mean'' design, and we find its compositional nature attractive. We also feel that CLASSIC's move to relax some of KANDOR's restrictions on expressive power is likely to increase CLASSIC's utility, and we applaud inclusion of such tools as co-reference constraints, forward chaining with rules, enumerated concepts, disjoint primitives, test concepts, host individuals and optional closed world semantics. CLASSIC's subsumption algorithm has ''low-order polynomial complexity'' and is thought, though not proven, to be complete. Although CLASSIC is not a radical departure from earlier systems, it does integrate many nice features in a satisfying way. Thus, we believe that CLASSIC is a neat synthesis of experience gained from its predecessors.

6.5 LOOM

LOOM, a descendant of NIKL, is now being produced at USC/Information Sciences Institute by MacGregor and his group. Like NIKL, LOOM possesses ''boxes'' for terminological and for assertional knowledge known as the TBox and UBox, respectively. When described by its creators in 1987, LOOM's major contribution was to be the introduction of two more boxes, one for *universal* knowledge and one for *default* knowledge, along with their associated reasoning components [MacGregor and Bates 87]. That LOOM architecture strove to provide a ''well-integrated collection of specialized reasoning components.''

The Universal Box, or UBox, extends the expressiveness of the T-Box.[27] Accordingly, a concept definition may include an *axiom* clause which conveys universal knowledge about the concept. In the UBox one can make statements about concepts which are merely necessary, e.g., ''Secure professors necessarily have tenure'', merely sufficient, e.g., ''All CUCS Professors are persons with private offices'', or both. One can also express non-definitional knowledge about concepts and relations such as coverings and disjointness, e.g.:

```
(defconcept Professor :primitive
  (:axioms (:disjoint-covering Assistant-Professor
                               Associate-Professor
                               Full-Professor)))
```

The TBox classifier does not take UBox information into account. Hence definitions, including those which introduce circular reference chains, can be made in the UBox and used for pattern matching and retrieval without impacting the TBox classifier. Another, more powerful UBox classifier acts on both TBox and UBox knowledge. As of this writing, it is unclear how much of the UBox classifier has been implemented.

The Default box is intended for representing assumptions. Default knowledge about a concept would be stated with a ''defaults'' clause in its definition. Some aspects of default knowledge to be addressed by LOOM in the short term are:

- LOOM might assume default role values for individuals in the absence of more explicit information. Of course, reclassification would become necessary if and when knowledge that contradicts the default assumptions is made known.
- LOOM may also allow some user control over its default reasoning behavior. For

[27] Which is used to state necessary and sufficient conditions regarding generic concepts.

example, one may specify open-world or closed-world semantics for an ABox knowledge base.

- Relations, which ordinarily have open-world semantics, may instead be given closed-world semantics, e.g., if the advisor relation is defined as follows and no instance of an advisor relationship has been asserted for a particular student, one may infer that the student has no advisor:

```
(defrelation Advisor
             :domain student
             :range professor
             :defaults closed-world-assumption)
```

Besides its general purpose reasoners, LOOM includes a variety of ‘‘narrow coverage’’ reasoners for such things as numbers, sets, intervals, and transitive and composite relations. These are realized as special kinds of concepts which appear in special sublattices of the concept taxonomy. Let’s look at a few examples.

A *set* concept may be defined by enumerating its instances, e.g.:

```
(defset Faculty-status (:values tenured untenured))
```

LOOM permits logical constants called *values* which need not be defined in the knowledge base. In the preceding example, **tenured** and **untenured** might be values.

An interval is a set which has predecessor and successor relations, e.g.,

```
(definterval Integer :primitive (specializes realnumber)
  (:values [-INFINITY..INFINITY])
  (:annotation
    (:membership-test (lambda (self) (integerp self)))
    (:predecessor-fn (lambda (self) (1- self)))
    (:successor-fn (lambda (self) (1+ self)))))
```

Finally, here are examples of composed and closure relations:

```
(defrelation intellectual-grandparent
             (:composition-of advisor advisor))

(defrelation intellectual-ancestor
             (:closure-of advisor))
```

A newer and significantly more ambitious view of LOOM as a knowledge representation facility that supports *model-driven programming* is proposed in [MacGregor and Brill 89]. We simply remark that LOOM will strive to combine three different programming paradigms:

1. Object-oriented programming, with operators and methods.

2. Data-driven programming with a production system and agenda-based control.

3. Constraint-based programming, with truth maintenance.

This ambitious approach is in direct opposition to a system like CLASSIC. It is our belief that such an integrated programming facility for knowledge representation could be extremely useful, and that LOOM’s design holds great potential. We eagerly anticipate its full realization.

6.6 K-Rep

K-Rep, currently under development at IBM's T.J. Watson Research Center, is designed in the spirit of KL-ONE with two major goals as points of departure [Mays 88, Mays *et al.* 91a]:

- To exploit inheritance and definitional classification in expert systems.
- To integrate multiple programming paradigms and special purpose reasoners in a transparent manner.

K-Rep is also noteworthy for its ability to manage an unusually large knowledge base in support of a very complex expert system, and for a number of carefully engineered facilities it provides to aid in that endeavor.

A K-Rep knowledge base is composed of generic concepts and their *instances*. The latter are analogous to KL-ONE's individual concepts. Roles attribute information to concepts, including value restrictions, values, and minimum and maximum cardinality. Role Value Maps (RVMs) specify arbitrary constraints among some roles of a concept. In addition, K-Rep supports facets on knowledge bases, concepts, roles and RVMs to describe their properties. A documentation facet may appear on any of them. Other facets of roles include default values, units of measure for values, and whether users may update the role interactively.

K-Rep allows roles to assume multiple values in the form of sets and bags whose elements are concepts. Ordered sets and bags are possible as well; the user specifies a Lisp predicate that orders the elements. Role values may also be time-valued sets indexed by (say) date. Time-valued sets might be used in a financial domain to record a series of price changes so that financial planning over a substantial period of time can reflect prices in effect at different times.

A role of a generic concept may specify, via a facet, that its value is obtained by *reverse inheritance* such that the corresponding role values of all the generic concepts it subsumes are aggregated into a set. This is useful from a knowledge engineering standpoint. For instance, the **cucs-professor** concept may have a **name** role with a value restriction of ***string***. Its value, by reverse inheritance from its subconcepts, may be viewed as an element of the set {Allen, Boult, Duchamp, ...}. By virtue of enforced semantics, the actual set value must be consistent with the value restriction.

Some roles have an attached *value function* which infers the role's value when the role is accessed at run time. The inferred value can then be validated against the role's value restriction. K-Rep provides transparent access to the inferred role values computed by the value functions. Inferred role values may optionally be *cached* for the purpose of preventing duplicate computations, that is, the value can be saved so that recomputation may not be necessary when the role value is accessed again. This feature is extremely useful when the computations are lengthy. K-Rep automatically maintains cache consistency. Whenever a role value changes, all inferred role values which depend on it become suspect and are discarded from the cache. Recomputations are done when an inferred role's value is accessed *iff* dependent values have changed during the interim. The cache mechanism allows independent problem solvers, implemented by means of value functions, to coexist without explicit communication.

K-Rep's Role Value Maps generalize the notion of role constraints in the KL-ONE family. An RVM consists of an arbitrary operator, including a user-defined one, rather than a predefined handful of operators such as equality and subset. Any number of role chains identify the arguments of an RVM operator, rather than just two. A null role chain may be used to designate the concept itself. Since RVMs are factored into concept classification, subsumption must be defined among operators and their role chains. RVMs in subordinate concepts may be further restricted via more specific operators, e.g., >= subsumes > and/or a more restrictive set of role chains with more restricted values, e.g., (= a b c) subsumes (= c a). In addition, K-Rep includes a constraint propagation mechanism tied to RVMs for the purpose of knowledge acquisition.

K-Rep's classifier organizes concepts into a definitional taxonomy according to their roles and RVMs, except that inferred roles are not considered during classification. Sometimes when a complete definition for a concept is not available, definitional classification is unable to place the concept into the proper taxonomic location. K-Rep supports rule-based heuristic classification as a means to supplement definitional classification of concepts in the face of incomplete information, particularly during problem solving. Explanations, consisting of reasons for and against the chosen location in the taxonomy, are entered as facets of the concept in question by the heuristic classifier's specialized inferencer.

K-Rep provides built-in semantics for several special types of concepts. For example, it can determine subsumption between arbitrary intervals on the real number line, or between date and time intervals specified in a wide variety of formats. Likewise, K-Rep handles subsumption for sets and bags.

Since K-Rep allows interactive editing of concepts, it also handles reclassification of any concept, be it instance or generic. Some care is taken to reclassify efficiently. Reclassification has implications for cache validity due to the fact that roles are inheritable. Because cached values may reflect very expensive computations, the strategy used to determine invalid cache entries must be precise.

K-Rep also includes a prototype facility to produce natural language explanations of taxonomic relationships resulting from classification [Weida 88]. These explanations aim to highlight relevant differences, if any, between a pair of concepts in clear, concise, cooperative English. This facility should prove helpful when a knowledge engineer who is debugging a knowledge base wants to determine why some concept failed to subsume some other concept.

Multiple knowledge bases can be used to partition the concept namespace, much as packages partition the Common Lisp namespace [Steele 84]. Knowledge bases are linked hierarchically to provide name inheritance and implicit shadowing of names. For instance, if one specifies that knowledge base KB1 *uses* KB2 and KB3, then concepts from KB2 and KB3 are visible in KB1, and a concept in KB1 may be defined in terms of concepts in KB2 or KB3. Individual knowledge bases can be created on the fly to explore a problem solving scenario. Furthermore, several KBs can be used for parallel consideration of alternative scenarios. An entire knowledge base can be deleted (along with any dependent knowledge bases, of course) without any need to check remaining knowledge bases for consistency. Thus scenarios which prove uninteresting can easily be discarded.

Much of the information available in machine-readable form today is stored in traditional databases. K-Rep provides an SQL interface so that a knowledge base can be populated by extracting information from an SQL relational database. In brief, tuples of a relation are mapped into instances of a generic concept such that role values correspond to column values.

K-Rep is now being applied in the context of FAME[28], an expert system to support marketing representatives in selling IBM mainframes [Kastner *et al.* 86, Apte *et. al.*]. FAME uses K-Rep to represent problems and problem-solving control as well as domain knowledge. In particular, FAME incorporates special-purpose problem solvers for equipment planning and financial planning. These problem solvers are defined in K-Rep and communicate through K-Rep. Much of this code consists of value functions for roles. Experience has shown that K-Rep's object-centered representation provides a useful framework around which to organize cooperative individual problem solvers.

We now return to K-Rep's goal of providing a clean representation and an organizational framework for substantial expert systems, which makes its outlook quite different from other systems in the KL-ONE mold. K-Rep's core embodies an enforced semantics, while value functions and facets provide hooks to Lisp. K-Rep therefore lacks formal semantics beyond its core and in principle its performance can be intractable. In practice this has not been a problem. Extensive use of value functions to encode expert systems can give K-Rep the flavor of a programming system for knowledge representation, in contrast to the ‘‘terminological service’’ nature of the other systems in this section. By striking a balance between the pure definitional taxonomy of, say, KL-ONE and the unrestricted environment of commercial systems such as KEE, we believe that K-Rep is well adapted to its task. In sum, we feel that K-Rep achieves the twin goals set forth at the beginning of this section.[29]

[28]For *Financial Marketing Expertise*.

[29]Having worked extensively on K-Rep, our opinion is not entirely independent.

7. Synthesizing Object-centered Representations and Reasoning Mechanisms

7.1 Motivation

Ideally, a single knowledge representation would serve all needs. We have seen that object-centered representations provide excellent support for terminological knowledge. Unfortunately, they are not intended for reasoning with assertional knowledge. At present, assertional reasoning is best done by theorem provers and rule-based systems which, conversely, are typically weak in terminological reasoning. Thus it makes sense to try to synthesize an object-centered representation with another representation so that the two can compensate for each other's weaknesses. Often this makes it possible to increase computational efficiency and extend representational coverage. There are costs involved in synthesizing different representations; they must be carefully weighed against the benefits. One may reasonably expect that more work will be involved in design, construction and maintenance of a knowledge base if more than one representation is involved. Special care must be taken to ensure that the components mesh well together by coordinating their activities, maintaining consistent information and minimizing redundancy. Beginning in section 7.3 we investigate a number of systems which have attempted to forge a useful synthesis.

7.2 Rule-based Systems

We now briefly introduce rule-based systems (or production systems) and argue that they can mesh neatly with object-centered knowledge representation. Rule-based systems are the dominant architectural paradigm for work in expert systems today and by extension, in artificial intelligence as well. They were introduced into AI research by Newell and Simon [Newell and Simon 72]. An excellent overview may be found in [Davis and King 78]. The primary components of a rule-based system are a working memory, a set of rules and an interpreter. Working memory consists of data structures which represent the system's current problem solving context, or state, and its contents are often viewed as a set of facts. Rules are composed of antecedents and consequents which relate a context to an action or conclusion. The interpreter repeatedly selects and applies rules until the problem is solved or no more progress is possible. When reasoning forward from one context to the next, the antecedents may be construed as conditions and the consequents as actions. The conditions form a pattern that is matched against working memory to determine if a rule is applicable in the current context. The actions specify changes in working memory. When reasoning backward to see if a hypothetical conclusion follows from known facts, the consequents may be taken as goals to be proven and the antecedents as recursive subgoals.

Frame systems (or semantic networks) provide support for rule-based systems. They can be used to represent rules as concepts by using roles for antecedents and consequents, as well as such ancillary information as author, purpose, and record of use. Rules can also operate on concepts in an object-centered representation so that the object-centered system provides the services of a working memory. Rules can test properties of frames and taxonomic relationships among frames in their antecedents and, in the case of forward-chaining, make assertions about concepts in their consequents. Observe that there is no need to write rules for taxonomic reasoning and inheritance of information when those inferences are provided by the frame system; rather rules can be written directly in terms of frame language queries and assertions. Prospector [Duda *et al.* 78], KEE [KEE User's Guide 88]and ART [ART Programming Tutorial 87] are examples of systems which represent their rules in the same object-centered form as the objects. ART, for example, achieves highly efficient run-time matching of rules by compiling them into a *rete* network [Forgy 82].

The antecedents of rules can be organized into a taxonomy so that their interrelationship is clear [Fikes and Kehler 85, Woods 86]. New antecedents can be introduced as variants of existing ones, leading to compact notation. If the taxonomy is definitional, a classification algorithm can be used to automatically place new rules in relation to existing ones. By viewing the set of rule antecedents as a taxonomy of recognizable problem solving situations, a more general, principled approach to knowledge engineering is encouraged. Then rules can be attached to the concept which describes the most general situation where they are applicable. Compared with traditional sets of independent rules, this approach is particularly beneficial when the number of rules grows large. In sum, the taxonomy supports partitioning and indexing of production rules so they can be used more effectively. An architecture for a general classification-based production system which follows these ideas is outlined in [Yen *et al.* 89]. They built a prototype using NIKL and planned to convert to LOOM.

Knowledge-based systems can be built on interpreters whose operation is guided by a set of frames. Some systems, especially those for diagnosis, operate by classifying the current problem solving situation and taking appropriate action. When problem solving situations are taxonomized as suggested above, a classification algorithm can efficiently select rules that match the current problem solving context. Notice that tests made once at a particular concept need not be repeated for its children. Alternatively, systems can be structured to reason from significant events signalled by invocation of attached procedures.

The CONSUL system handled requests and provided explanations as an interface between a user and a software system such as electronic mail [Mark 81]. CONSUL used rules to represent relationships among users, services and software systems. Requests are treated as descriptions which are repeatedly classified and then reformulated by rules associated with the current classification until an executable interpretation results. If this is not possible, a similar approach is used to produce an explanation. Thus CONSUL can arrive at an appropriate response. Note that while CONSUL employed rules within a classification framework as we have outlined above[30] its operation was more constrained than a general production system.

Rule-based inference can contribute to the operation of a frame system itself. For instance, attached procedures can be realized as sets of rules to be executed. Also, rules are readily used to conclude that sufficient conditions for class membership hold. This approach was taken in Centaur, a medical consultation system concerned with diagnosing pulmonary problems [Aikins 83, Aikins 84]. It can easily be employed in domain-independent object-centered systems, e.g., KEE [Fikes and Kehler 85]. As noted in section 6.6, K-Rep's automatic classification algorithm, which operates on concepts defined in terms of necessary and sufficient attributes for class membership, can also be supplemented by rules which test only sufficient conditions.

---

[30]Not all of this is described in the literature.

We now turn to detailed descriptions of several hybrid systems which couple definitional taxonomies with theorem provers or rule systems for knowledge representation and reasoning.

7.3 KRYPTON

KRYPTON is a hybrid knowledge representation system that integrates a frame based terminological component with a logic based assertional component [Brachman *et al.* 83a, Brachman *et al.* 83b, Brachman *et al.* 85]. Along with its contemporary, KL-TWO, which is described in the next section, KRYPTON diverged from earlier hybrid systems by maintaining a clear distinction between these two types of components, viewing them as serving entirely different needs. Both systems grew out of Brachman's work on KL-ONE and their common ancestry accounts for noticeable similarities. They integrate the strength of KL-ONE style systems at forming complex descriptions with the facility of FOPC for representing incomplete information. The most important contribution of KRYPTON is the notion of a purely functional interface to a knowledge representation facility.

KRYPTON's terminological component is referred to as the TBox. It allows one to define concepts and roles similar to those of KL-ONE. The TBox is used exclusively to express structured descriptions of terms and to reason about their taxonomic relationship; TBox expressions carry no assertional force. Taxonomic reasoning determines subsumption and disjointness relationships among terms. KRYPTON's designers recognized a need for additional TBox reasoning capabilities but apparently did not pursue the matter.

KRYPTON's assertional component is known as the ABox. It permits one to describe and reason with domain theories. The language of the ABox amounts to first-order predicate calculus and we will not detail it further.

The TBox and ABox are cleanly connected in KRYPTON. TBox terms serve as definitions of ABox predicates, thereby establishing a structural relationship among ABox predicates. This quality differentiates the ABox from standard first order logic systems whose predicate names are both independent and primitive.

As mentioned earlier, the user interface to KRYPTON is strictly functional. That is, a set of functions for interacting with the TBox and the ABox are provided, and they completely specify KRYPTON's knowledge representation service. KRYPTON provides TELL operations to add knowledge and ASK operations for queries, plus a symbol table which provides the names of TBox terms to the user for reference. The user is not permitted to access or modify the content of the boxes in any other way. Consequently, KRYPTON offers the user a knowledge level [Newell 82] account of the service it provides which is separate from its symbol level implementation. An interesting consequence is that the system need not explicitly remember everything it is told as long as it produces correct responses to queries. The knowledge level view of KRYPTON is elaborated in [Brachman *et al.* 85].

As far as the TBox is concerned, TELL defines a name by associating it with a concept or role expression. The TBox ASK operation supports two kinds of yes/no questions:

- Does term1 *subsume* term2?
- Is term1 *disjoint from* term2?

The TELL operation on the ABox asserts that a certain ABox sentence is true. The ABox's ASK operation finds out whether a given sentence is true or false[31] according to the current theory. This determination takes into account the terminology of the sentence as it is defined in the TBox. While the TBox/ABox interaction could be accomplished by an implementation which asserts sentences in the ABox corresponding to each definition in the TBox and performs standard inferences on the expanded ABox knowledge base, that approach would be extremely redundant. Instead, KRYPTON extends ABox inference rules to work directly with inter-predicate dependencies derivable from TBox definitions by extending the meaning of unification. Importantly, the extended unification is done in a computationally efficient manner. For details, refer to [Brachman *et al.* 85].

KRYPTON's introduction of a purely functional interface to a KR system is a significant contribution, and its early exploration of a sound TBox/ABox interaction is noteworthy. It seems that KRYPTON never got beyond the experimental prototype stage, and it remains unclear, at best, whether KRYPTON's hybrid reasoning service was on its way to becoming a practical tool for serious applications.

7.4 KL-TWO

KL-TWO is a hybrid representation system developed by Vilain and his colleagues at BBN that combines two complementary reasoners: PENNI, a propositional reasoner, and NIKL (Section 6.2), a terminological reasoner [Vilain 85]. KL-TWO offers both forward and backward reasoning. It is distinguished from earlier hybrids by PENNI's restricted language which stands in contrast to the first order logic theorem provers used previously. The choice of PENNI was motivated by a desire to trade some expressiveness for a high degree of efficiency.

PENNI supports a database of propositional assertions expressed in predicate calculus with equality but without any quantification. These assertions are managed by a truth maintenance system which performs dependency-directed backtracking and provides an efficient mechanism for deduction. Among other things, PENNI allows *if-added* and *if-needed* demons.

NIKL's language, like PENNI's, is restricted, and therefore amenable to efficient implementation. As we have seen earlier, NIKL's concept definitions can be viewed as sets of universally quantified sentences. In KL-TWO, NIKL's concept names correspond to PENNI's predicate names. Thus, NIKL can extend PENNI's expressive power by providing limited quantificational reasoning. KL-TWO achieves this synthesis by instantiating NIKL sentences in PENNI at appropriate times in support of both forward reasoning and backward reasoning.

The forward reasoner is invoked whenever new propositions concerning an individual are asserted in PENNI. Its purpose is to discover which NIKL sentences are applicable to that individual. The forward reasoner creates and classifies a NIKL concept, referred to as a *most specific generalization (MSG)*, which abstracts all the PENNI propositions concerning the individual. Then, the NIKL sentences applicable to the individual are just those which express subsumption relations between its MSG and other concepts represented in NIKL. These sentences can be instantiated as PENNI propositions on an as-needed basis. Upon instantiation,

[31]Or perhaps its truth value is unknown.

demons may be triggered by the proposition's predicate name.

The backward reasoner can use NIKL to help answer queries about individuals by applying NIKL sentences concerning the individual. As an example, Vilain considers a query of the form (C I), where I is an individual and C is a predicate name for which a corresponding concept has been defined in NIKL. The queried proposition is true if C subsumes the MSG of I.

KL-TWO demonstrates that two restricted, complementary reasoners can be combined in a way that is both useful and computationally efficient. On the other hand, the combination still suffers from limited expressiveness in that KL-TWO does not directly support quantification beyond NIKL's limited form of universal quantification. As Vilain points out, it is quite difficult to determine if and when to instantiate quantified sentences without the benefit of domain specific knowledge. Like earlier hybrid reasoners, KL-TWO simply provides hooks for this purpose. Application-specific quantification can be encoded with domain dependent reasoners using PENNI's demon facility.

7.5 KROPS

IBM's KROPS is a fusion of K-Rep, an object-centered knowledge representation described in section 6.6, and YES/OPS[32], a variant of the OPS5 rule-based programming language [Daly *et al.* 88].[33]YES/OPS is presented in [Schor *et al.* 86]. An interesting aspect of this combination is that both K-Rep and YES/OPS are useful in their own right, and in fact, were developed independently.

Members of the OPS family of languages are widely used in expert systems work. They offer great flexibility in pattern matching and in control of the production system. OPS systems directly support forward chaining by repeatedly matching rules against working memory, selecting a rule and executing it. Working memory is composed of working memory elements (WMEs) made up of attribute/value pairs. A notable feature of OPS languages is the *rete*, a network data structure that efficiently records partial match information for rule antecedents and is incrementally updated to reflect changes in working memory. YES/OPS enhancements to the OPS formalism include pattern matching in the consequents of rules and the possibility of using arbitrary lisp expressions anywhere in a rule.

From the perspective of K-Rep, KROPS adds a powerful inferencing capability with production rules that match arbitrary patterns against K-Rep concepts. Conversely, KROPS extends YES/OPS with the facility for abstraction and aggregation of K-Rep's definitional taxonomy. This is beneficial because, as we have noted, human experts tend to think in terms of abstractions. In contrast with standard OPS languages, a knowledge engineer can write rules directly in terms of generic concepts which represent many individuals. Inferencing therefore takes place at the desired level of generality, possibly yielding significant gains in computational efficiency. The knowledge engineer may rely on K-Rep's automatic reclassification feature to guarantee a consistent network.

---

[32]Now known as ECLPS and offered as an IBM program product.

[33]An introduction to OPS5 is [Brownston *et al.* 85].

In KROPS, features of K-Rep and YES/OPS may be freely intermingled. Rules can operate on concepts in K-Rep's inheritance network as well as elements of YES/OPS' working memory. Pattern matching capabilities and syntax are quite similar in both cases.

When deciding how to represent particular knowledge, the choice between concepts and working memory elements hinges on how the knowledge will be used. Although concepts are better suited to the notions of abstraction and aggregation, maintaining the concept network in a consistent state via reclassification imposes substantial computational overhead on the system. Working memory elements can be used effectively for control information and temporary variables which need not be classified, yet pattern matching on them is still available. Also, in this way domain knowledge represented in concepts is kept separate from other kinds of information.

Working memory elements and concepts coexist in the rete. WMEs, represented as vectors, may be accessed very efficiently but their information content is fixed. Concepts are far less efficient because value restrictions of roles are looked up at run time. On the other hand, they are more flexible because they support late binding of values. It would be interesting to see how classification could be integrated with the rete, however this matter remains unexplored.

KROPS was developed as a practical vehicle for FAME and was used extensively at one point. This gives us some evidence of KROPS' viability. However the KROPS implementation never became a polished product. Lack of a computationally efficient integration between the concept taxonomy and the RETE network is a major handicap.

8. Other Approaches to Hybrid Reasoning With Frames

For the sake of contrast we present three additional approaches to the integration of frames with assertional reasoning mechanisms. Unlike the systems discussed earlier, these systems do not employ a classification algorithm to maintain a definitional taxonomy. Charniak's system treats frames as a convenient notation for some of the predicate calculus statements that are reasoned with by a theorem prover [Charniak 81]. Prospector applies partitioned semantic networks to the representation of both terminological information and rules [Duda *et al.* 78]. Moreover, Prospector's reasoning is guided by an *inference network* consisting of connections among nodes in the semantic network. Finally, Centaur is a tightly coupled system of frames and rules for the task of diagnosis [Aikins 83, Aikins 84].

8.1 Charniak's System

Eugene Charniak proposed and implemented a combination of frames and predicate calculus that addressed the sometimes divergent requirements of problem solving and language comprehension [Charniak 81]. Charniak explicitly viewed frames as a convenient shorthand for predicate calculus statements. His system was redundant in the sense that it used *both* notations for the same information.

We have already examined arguments concerning the relative merits of frames and predicate calculus. Here we mention some of Charniak's perspective on these issues in the settings of natural language understanding and problem solving. Note that for him, frames are used as a collection of stereotypical knowledge. In the case of natural language comprehension:

1. Frames partition knowledge in a natural way so that only appropriate knowledge is activated in a given situation.

2. Frames represent inferential knowledge in a natural way.

For problem solving:

1. The correctness of a solution is crucial and uncertainty over the semantics of frames is a major liability.

2. Predicate calculus is more amenable to piecemeal knowledge acquisition.

3. To solve a problem with conjunctive subgoals, steps in the solutions of the subgoals may need to be interspersed. This seems to go against the grain of the frame-based methodology.

Charniak's frames function as abbreviations for precisely specified predicate calculus statements. By expanding the abbreviations, the system makes explicit what is generally left implicit in frame systems. We illustrate our informal discussion with the following frame, where symbols beginning with ‘‘?’’ are variables.

```
[professor
        isa: (person ?professor)
      slots: (rank (professor-rank ?rank))
             (office (professor-office ?office))
      facts: (degree ?professor Ph.D.)
             (occupies ?professor ?office)
             ...]
```

Such a frame abbreviates facts that are true of all its instances. Now let us see how the abbreviations are expanded to make their meaning explicit, with the caveat that in the implementation they take a different form. Consider inferential facts. The facts in the **professor** frame are only true if ?professor is indeed bound to a professor. The first fact shown, for example, is an abbreviation for:

```
FORALL (?professor (professor ?professor))
  [degree ?professor Ph.D.]
```

Therefore, ?professor is effectively a typed variable.

Slots are viewed as variables which are existentially quantified over frame instances. The fact that a professor occupies an office is made explicit as follows:

```
FORALL (?professor (professor ?professor))
  EXISTS (?office (professor-office ?office))
    [occupies ?professor ?office]
```

Inheritance is supported via is-a, but cancellation of inheritance is not permitted.

Frames also constrain the form of simple predicate calculus statements, i.e., those without connectives. Simple PC statements must all be *frame-instance statements* such as (professor Kender) or *slot-filler-statements* such as (rank Kender associate-professor). This enables detection of errors such as illegal predicates. In addition, frame-instance statements take exactly one argument[34]and slot-filler statements take exactly two. Furthermore, (office Kender CSB467) is legal with respect to its first argument on the condition that Kender names a **professor** frame whereas (office CSB467 Kender) would be illegal because CSB467 does not name a professor. Similar type checking obviously applies to the second arguments of slot-filler statements.

Examples such as the **professor** frame could well be used in a natural language understanding system. Charniak also suggests how a particular problem solving language, Micro-NASL, might be cast in his frame format. The interested reader is referred to [Charniak 81].

Charniak's system uses frames as organizing tools to collect related information. Any general predicate calculus statement must be attached to frame. Statements about particular objects (introduced in a story, perhaps) are also attached to a particular frame, however they go into a conceptually separate database. A particular frame, e.g., for Kender, is activated if and only if a statement of the form (professor Kender) is entered in the database. When a frame is activated, all frames above it in the is-a hierarchy are activated as well.

The reasoning component of Charniak's system is a deductive theorem prover with forward and backward chaining. It is described in his well known AI programming text [Charniak, Riesbeck and McDermott 87].

In sum, Charniak's system has several strong points. He endows the ‘‘natural’’ frame representation with a precise meaning in predicate calculus. The predicate calculus representation retains its ‘‘fine fact granularity’’. Frames serve to extend predicate calculus with

[34]Actually, they may also take fillers of the frame's required slots.

partitioning of facts, a set of convenient abbreviation conventions and derived constraints on predicates and arguments. The redundant notations, on the other hand, are less desirable.

8.2 Prospector

Prospector is a rule-based system constructed in the mid-seventies at SRI which was designed to act as a consultant for geologists in the field of mineral exploration [Duda *et al.* 78]. Prospector uses Hendrix' partitioned semantic network formalism to represent the constituents of its rules [Hendrix 79]. There is a large terminological network composed of *s arcs* which indicate subset relations and *e arcs* which convey element-of relations. The idea was that the semantic network would explicate the structure of the problem domain while the rules would represent judgemental knowledge. Recall that partitioned networks extended the strong points of semantic networks with the full expressive power of predicate calculus.

A rule in Prospector takes the form of an implication whose antecedent is a logical function of individual pieces of evidence and whose consequent is a hypothesis supported by the evidence. As shown in Figure 8-1 taken from [Duda *et al.* 78], some particular rule, **R**, is an element of **Rules**. Separate partitions are allocated for its antecedent and consequent. Figure 8-2, taken from the same source, shows the rule ‘‘barite overlying sulfides suggests the possible presence of a massive sulfide deposit.’’

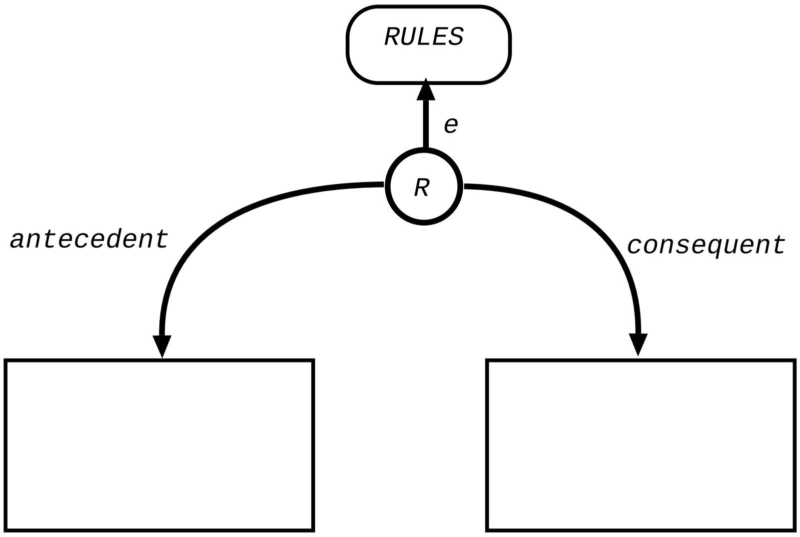

Figure 8-1: Prospector Rule Form [Duda *et al.* 78]

Since the antecedents and consequents of each rule are partitions within Prospector's semantic network, they are naturally linked together to form an *inference network*. Rules are linked explicitly when the consequent of one rule is antecedent to another. Information propagates along explicit links via rule chaining. Rules are also linked implicitly by chains of element and/or subset links, so information propagates through these links as well.

Observe that links among rules represent *potential* inferences. The nodes used in the

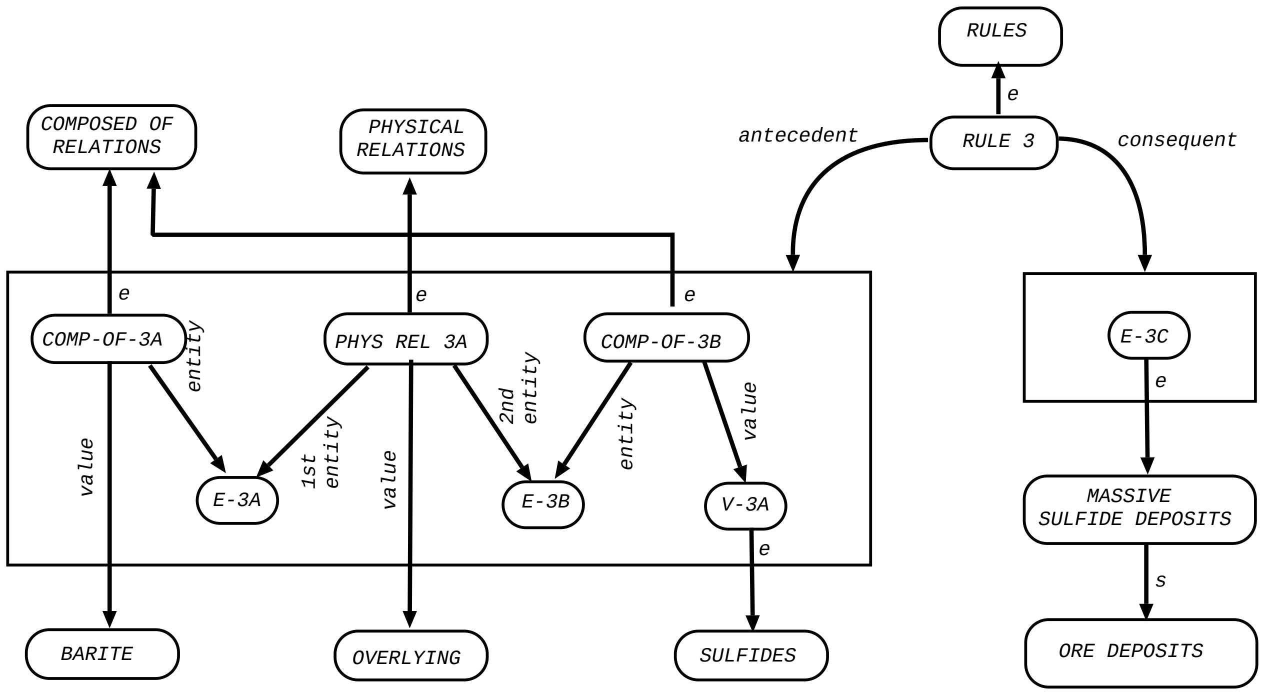

Figure 8-2: Prospector Rule Example [Duda *et al.* 78]

representation of a rule constitute variables which can be bound in various ways. Therefore, links among rules are actually realized dynamically during a consultation by a pattern matching process.

In Prospector's domain, both the evidence and the rules entail a degree of uncertainty. An important task for Prospector is the propagation of probability estimates in the inference network. A subjective probability is associated with each relation. For example, if ''Entity-1 is composed of rhyolite with probability P1'' then P1 is attached to the *composed-of* relation. With rules, the probability of the antecedent is computed from its constituents by recursive use of Zadeh's fuzzy set formulas. Then a modified form of Bayes' rule is used to compute the probability of the consequent. Whenever a probability changes because new information has been entered, this procedure propagates the effects of the change through the network.

Prospector's performance was evaluated by applying it to actual geological sites. As a result, Prospector has been judged a great success, and it is often regarded as one of the most significant success stories in the annals of artificial intelligence. This strongly validates Prospector's approach to knowledge representation, including its combination of rules with semantic networks. By adopting Hendrix' partitioned network formalism, with all the representational power of first order logic, Prospector contrasts sharply with systems using the restricted expressiveness of KL-ONE style technology. Although Prospector's expressiveness holds the potential for intractable worst-case performance, this was evidently not an insurmountable problem in practice.

8.3 CENTAUR

Centaur was built by Aikins as an alternative to PUFF, an earlier system which used a production rule formalism[35] to interpret data from pulmonary function tests and make a diagnosis. Centaur combines frames and production rules in a hypothesis-driven consultation system whose processing is organized around its knowledge of prototypical situations [Aikins 83, Aikins 84]. Frames represent prototypical cases while rules represent more specific inferential knowledge. Centaur's task-oriented integration of frames and production rules contrasts with the general-purpose design of KROPS.

Centaur's *prototypes* are hierarchically related frames, with inheritance, which represent classes of hypotheses corresponding to typical patterns of data associated with pulmonary diseases. A high level **Consultation** prototype represents the consultation process itself. It has slots corresponding to the stages of a consultation. Prototypes focus Centaur's processing, constrain it to elicit relevant information from the user, provide an organizing framework for rules and an explicit context for rule invocation, and aid detection of inconsistent or erroneous data. A sample partial instantiation of a prototype for obstructive airways disease (OAD), taken from [Aikins 83], is shown in the upper portion of Figure 8-3.

Author: Aikins
Date: 27-OCT-78 17:13:29
Source: Fallat
Pointers: (degree MILD-OAD) (degree MODERATE-OAD) ...
(subtype ASTHMA) (subtype EMPHYSEMA) ...
Hypothesis: There is Obstructive Airways Disease.

If-confirmed: Deduce the degree of OAD
Deduce the subtype of OAD
Action: Deduce any findings associated with OAD
Print the results associated with OAD

Fact-Residual Rules: RULE157, RULE158, ...
Refinement Rules: RULE036, RULE038, RULE039, ...
Summary Rules: RULE053, RULE054, RULE055, RULE083, ...

Components:

Total Lung Capacity:	**Plausible Values:** >100
	Importance Measure: 4
Reversibility:	**Inference Rules:** RULE019, RULE020, RULE022, RULE025
	Importance Measure: 0

Figure 8-3: Centaur prototype and components [Aikins 83]

---

[35]namely Emycin.

A prototype contains a number of other frames called *components* which represent its important domain features, e.g., characteristic aspects of pulmonary disease such as total lung capacity. Here are some slots that may appear within components:

- Plausible values
- Possible error values
- Default value
- Importance measure
- Inference rules

Two sample components are shown in the lower portion of Figure 8-3. The inference rules (which are only named in the figure) have premises which match ranges of values for some of the prototype's components. They produce a value for the component. For example, one rule might conclude that an obstructive airways disease is in fact reversible. Notice that only a restricted set of rules apply to computing the value of a component and that they operate in a relatively narrow context. Plausible values and possible error values are really condition-action pairs whose actions may involve making a conclusion about the values of other components or triggering another prototype. For instance, a total lung capacity exceeding 100 characterizes persons suffering from OAD.

Prototypes also have a variety of domain-independent slots. General information slots include static information: documentation such as *author*, *date* and *source*, English phrases for user interaction, such as *explanation* and *hypothesis,* and pointers to other prototypes, such as *more general*, *more specific*, *degree*, *subtype* and *alternate*. Dynamic information slots record the certainty that the prototype matches the data and, for purposes of explanation, the circumstances under which the prototype was invoked.

Control slots contain Lisp clauses that Centaur invokes at suitable points to control the consultation. The definition of the high level **Consultation** prototype determines when control slots are invoked. There are four types of control slots: *to-fill-in*, *if-confirmed*, *if-disconfirmed* and *action*. The latter is run after final conclusions about the prototype have been derived.

There are three types of rule slots in a prototype. They are also invoked at specific times during a consultation. *Fact residual* rules try to account for left over facts following a conclusion. *Refinement rules* may suggest additional tests and such. *Summary rules* summarize the prototype's information content. In addition, there are *triggering rules,* associated with individual parameters of components, whose premises match values of components and whose actions suggest alternative hypotheses.

Centaur follows an agenda-driven control regimen which reflects hypotheses represented by the prototypes. The system tries to match one or more prototypes with the patient's case. There is always one current prototype being matched, beginning with the top level consultation prototype. Centaur has a simple interpreter to carry out general system control. Context-specific control is contained in the control slots of individual prototypes. In [Aikins 83] a synopsis of the consultation process is given.

In Centaur, context and control are represented explicitly in frames: prototypes and components group rules according to situations where they apply and further categorize them according to when they are relevant. This means that ‘‘the expert can specify what to do in a given context.’’ Modifications can be localized and their effects predicted relatively easily. Because Centaur’s rules need not express context and control information, they convey only chunks of medical expertise.

PUFF exhibited a number of problems common to production rule systems and numerous improvements can be cited in Centaur’s approach. Prototypes support control flow tailored to context such that the progress of a consultation corresponds to a physician’s reasoning. This aids explanation. Questions are asked if and when they are relevant. Prototypes are able to represent expected patterns of data. This makes it easy to detect inconsistent or erroneous data. A drawback of Centaur is the rather complex, task-oriented nature of its architecture.

8.4 Conclusion

This section has examined three different hybrid knowledge representation systems, none of which fits the KL-ONE style. Of these, Charniak’s system is closest to hybrids based on the KL-ONE family such as KL-TWO and KRYPTON. Exceptions to inheritance are forbidden, so Charniak’s taxonomy is definitional, though his system does not attempt automatic classification. Charniak’s system employs frames as a convenient shorthand for collections of predicate calculus statements, thereby imbuing the frames with a clean logical basis. His system includes a theorem prover which acts on information expressed in the frame formalism, while the frames, in turn, constrain the permissible form of simple PC statements. A shortcoming of Charniak’s approach is the redundancy between the two notations. Hendrix’ partitioned networks, as employed in Prospector, also adopt an explicit predicate calculus interpretation. Whereas the form of PC statements expressible via Charniak’s frame notation is quite limited, partitioned networks support the full power of predicate calculus. Furthermore, Prospector’s semantic network directly embodies an inference network that guides Prospector’s problem-solving operation, yielding a single, integrated notation. This architecture allowed Prospector to maintain acceptable performance levels despite the potential intractability inherent in its broad representational power. Prospector is also distinguished from the other systems we considered by its built-in facility for probabilistic reasoning. Centaur, tailored for diagnosis problems, trades generality for performance. It is the most task-oriented system we have looked at, both in the format of its frames and in its explicit representation of control for the diagnostic task. Centaur also proved highly effective, but it has significantly more ad-hoc flavor than the other systems. In conclusion, it does not seem possible to conclude that any of these three approaches is categorically better or worse that the KL-ONE style approach. Rather, it is necessary to carefully consider the problem at hand when making a choice.

9. Conclusion

In this paper we have considered a particular class of taxonomic knowledge representation systems which are distinctive because they forbid exceptions to inheritance. By adopting this strong stance, a system can provide definitions rather than just defaults, and its taxonomy can be organized and partitioned in an unambiguously prescribed manner. Furthermore, a definitional taxonomy admits a type of inference called classification, which automatically places new definitions into the taxonomy at the uniquely appropriate location. Classification also provides support such as type checking and pattern matching. While definitional taxonomies are significantly limited by the omission of exceptions, that is an unavoidable cost of the purely definitional approach. The knowledge representation community is sharply divided as to the merit of this exchange and the proper role of definitional classification in knowledge representation.

We find that definitional taxonomies of structured objects are particularly good when it comes to defining terminological knowledge of concepts. Current work seeks to extend the capabilities of taxonomic reasoners with built-in semantics for specialized types of concepts. Definitional taxonomy systems are not adept with assertional knowledge. Fortunately, these strengths and weaknesses complement those of other approaches, such as production systems, theorem provers and object-oriented programming. In recent years, this observation has led to development of numerous hybrid systems. Although such combinations seem promising, we have seen that a great deal of work remains to achieve a smooth, powerful, yet economical combination of reasoners with clear semantics.

Present day definitional taxonomy systems have been forced to sacrifice logical expressiveness (e.g., no negation or disjunction and limited quantification) for computational efficiency. Still, most of them remain computationally intractable in the worst case. Much current work seeks to analyze the complexity of various language formulations and to identify good tradeoffs between expressiveness and tractability. In our opinion, the emphasis should be placed on increased expressiveness as this would make such systems more widely applicable in practice. One exciting possibility for boosting tractable performance at the expense of some precision is the notion of vivification, outlined in section 2.7. An issue related to expressiveness and tractability is the completeness of subsumption algorithms. An important direction for future research is to determine how complete existing algorithms are and how much completeness is required, as well as to come up with provably complete algorithms for expressive languages.

We believe that another major limitation of present day approaches is their inability to scale up in size and scope. Today one generally finds rather small, very domain-specific knowledge bases that are maintained in text files at the source code level, loaded into main memory *en toto* and operated on by a single process. In time, we expect to see extremely large knowledge bases with much broader scope that are created, maintained and used by numerous individuals over a long period of time and over a wide geographic area, with services analogous to those provided by today's database management systems. Thus, tomorrow's knowledge base management systems (KBMS) must support persistence and sharing of knowledge at a suitable level of granularity, with distributed access and concern for matters of integrity and security. Work in this area has begun at IBM's T. J. Watson Research Center [Mays *et al.* 91b] and at MCC [Lenat *et al.* 86].

References

[Abrett *et al.* 87] Abrett, G., Burstein, M. Gunshenan, A. and Polanyi, L.
KREME: A User's Introduction.
BBN Report No. 6508, Bolt Beranek and Newman, Cambridge, MA, 1987.

[Abrett and Burstein 87]
Abrett, G., Burstein, M.
The KREME Knowledge Editing Environment.
International Journal of Man-Machine Studies 27:103-126, 1987.

[Aikins 83] Aikins, J. S.
Prototypical Knowledge for Expert Systems.
Artificial Intelligence 20(2):163-210, 1983.

[Aikins 84] Aikins, J.
A Representation Scheme Using Both Frames and Rules.
Rule-Based Expert Systems.
Addison Wesley, Reading, MA, 1984, pages 424-440.

[Apte *et. al.*] Apte, C., Dionne, R., Griesmer, J., Karnaugh, M., Kastner, J., Laker, M. and Mays, E.
An Experiment in Constructing an Open Expert System using a Knowledge Substrate.
IBM Journal of Research and Development.

[ART Programming Tutorial 87]
Clayton, B. D.
ART Programming Tutorial.
Inference Corp.
1987
Version 3.0.

[Baader 90] Baader, F.
Terminological Cycles in KL-ONE-based Knowledge Representation Languages.
In *Proceedings of AAAI-90*, pages 621-626. American Association of Artificial Intelligence, Boston, MA, 1990.

[Balzac 86] Balzac, S. R.
A System for the Interactive Classification of Knowledge.
Master's thesis, MIT, 1986.

[Barr 80] Barr, A.
The Representation Hypothesis.
Working Paper HP-80-1, Heuristic Programming Project, Stanford University.
January, 1980
Talk given at IJCAI-79, Tokyo.

[Barr and Davidson 81]
Barr, A. and Davidson, J.
Knowledge Representation.
The Handbook of Artificial Intelligence.
William Kaufman, Los Altos, CA, 1981, pages 141-222.

[Beck *et al.* 89] Beck, H. W., Gala, S. K., and Navathe, S. B.
Classification as a Query Processing Technique in the CANDIDE Semantic Data Model.
In *Proc. Fifth International Conference on Data Engineering*, pages 572-581. Los Angeles, CA, 1989.

[Bobrow 75] Bobrow, D. G.
Dimensions of Representation.
Representation and Understanding: Studies in Cognitive Science.
Academic Press, New York, NY, 1975, pages 1-34.

[Bobrow and Winograd 77]
Bobrow, D. G. and Winograd, T.
An Overview of KRL, A Knowledge Representation Language.
Cognitive Science 1(1):3-46, 1977.
reprinted in *Readings in Knowledge Representation*.

[Bobrow and Winograd 79]
Bobrow, D. G. and Winograd, T.
KRL: Another Perspective.
Cognitive Science 3(1):29-42, 1979.

[Borgida *et. al.* 89]
Borgida, A., Brachman, R. J., McGuinness, D. L. and Resnick L. A.
CLASSIC: A Structural Data Model for Objects.
In *Proc. 1989 ACM SIGMOD International Conference on the Management of Data*, pages 58-67. ACM SIGMOD, Portland, OR, 1989.

[Brachman 79] Brachman, R. J.
On the Epistemological Status of Semantic Nets.
In N. V. Findler (editors), *Associative Networks: Representation and Use of Knowledge by Computers,* pages 3-50. Academic Press, New York, 1979.

[Brachman 83] Brachman, R. J.
What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks.
IEEE Computer 16(10):30-36, 1983.

[Brachman 85] Brachman, R. J.
"I Lied About the Trees" (or, Defaults and Definitions in a Knowledge Representation).
AI Magazine 6(3):80-93, 1985.

[Brachman 88] Brachman, R. J.
The Basics of Knowledge Representation and Reasoning.
AT&T Technical Journal 67(1):7-24, 1988.

[Brachman *et al.* 85]
Brachman, R. J., Gilbert, V. P. and Levesque, H. J.
An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of Krypton.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 532-539. Los Angeles, CA, 1985.

[Brachman *et al.* 83a]
Brachman, R. J., Fikes, R. E. and Levesque, H. J.
KRYPTON: Integrating Terminology and Assertion.
In *Proceedings of AAAI-83*, pages 31-35. American Association of Artificial Intelligence, Washington, D.C., 1983.

[Brachman *et al.* 83b]
Brachman, R. J., Fikes, R. E. and Levesque, H. J.
KRYPTON: A Functional Approach to Knowledge Representation.
IEEE Computer 16(10):67-73, 1983.

[Brachman and Levesque 84]
Brachman, R. J. and Levesque, H. J.
The Tractability of Subsumption in Frame-Based Description Languages.
In *Proceedings of AAAI-84*, pages 34-37. American Association of Artificial Intelligence, Austin, Texas, 1984.

[Brachman and Levesque 85]
Brachman, R. J. and Levesque, H. J. (editors).
Readings in Knowledge Representation.
Morgan Kaufmann, Los Altos, CA, 1985.

[Brachman and Schmolze 85]
Brachman, R. J. and Schmolze, J. G.
An Overview of the KL-ONE Knowledge Representation System.
Cognitive Science 9(2):171-216, 1985.

[Brownston *et al.* 85]
Brownston, L., Farrell, R., Kant, E. and Martin, N.
Programming Expert Systems in OPS5.
Addison Wesley, Reading, MA, 1985.

[Charniak 81] Charniak, E.
A Common Representation for Problem-Solving and Language Comprehension Information.
Artificial Intelligence 16(3):225-255, 1981.

[Charniak, Riesbeck and McDermott 87]
Charniak, E., Riesbeck, C. K., McDermott, D. V. and Meehan, J. R.
Artificial Intelligence Programming, 2nd. ed.
Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[Daly *et al.* 88] Daly, T., Kastner, J. and Mays, E.
Integrating Rules and Inheritance Networks in a Knowledge-Based Financial Marketing Consultation System.
In *HICSS-21*. Hawaii, January, 1988.

[Davis and King 78]
Davis, R. and King, J.
An Overview of Production Systems.
Machine Intelligence 8.
Ellis Horwood, Chichester, England, 1978, pages 75-82.

[Devanbu *et al.* 89]
Devanbu, P., Selfridge, P. E., Ballard, B. W., and Brachman, R. J.
A Knowledge-Based Software Information System.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 110-115. Detroit, 1989.

[Doyle and Patil 89]
Doyle, J. and Patil, R.
Two Dogmas of Knowledge Representation: Language Restrictions, Taxonomic Classification, and the utility of representation services.
Technical Report MIT/LCS/TM-387.b, MIT/Laboratory for Computer Science, Cambridge, MA, 1989.

[Duda *et al.* 78] Duda, R. O., Hart, P. E., Nilsson, N. J. and Sutherland, G. L.
Semantic Network Representations in Rule-Based Inference Systems.
Pattern Directed Inference Systems.
Academic Press, New York, NY, 1978, pages 203-221.

[Etherington *et al.* 89]
Etherington, D. W., Borgida, A., Brachman, R. J. and Kautz, H.
Vivid Knowledge and Tractable Reasoning: Preliminary Report.
In *Proceedings of the International Joint Conference on Artificial Intelligence,* pages 1146-1152. Detroit, 1989.

[Fikes and Kehler 85]
Fikes, R. and Kehler, T.
The Role of Frame-Based Representation in Reasoning.
Communications of the ACM 28(9):904-920, 1985.

[Fillmore 68] Fillmore, C. J.
The Case for Case.
Universals in Linguistic Theory.
Holt, Rinehart and Winston, New York, 1968, pages 1-88.

[Finin 86a] Finin, T.
Understanding Frame Languages (Part I).
AI Expert :44-50, November, 1986.

[Finin 86b] Finin, T.
Understanding Frame Languages (Part II).
AI Expert :51-56, December, 1986.

[Finin and Silverman 86]
Finin, T. and Silverman, D.
Interactive Classification as a Knowledge Acquisition Tool.
Expert Database Systems.
Benjamin/Cummings, Menlo Park, CA, 1986, pages 79-90.

[Forgy 82] Forgy, C. L.
Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem.
Artificial Intelligence 19(1):17-37, 1982.

[Fox 79] Fox, M. S.
On Inheritance in Knowledge Representation.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 282-284. Tokyo, 1979.

[Greiner and Lenat 80]
Greiner, R. and Lenat, D. B.
A Representation Language Language.
In *Proceedings of AAAI-80*, pages 165-169. American Association of Artificial Intelligence, Stanford, CA, 1980.

[Hayes 77] Hayes, P. J.
In Defence of Logic.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 559-565. Cambridge, MA, 1977.

[Hayes 79] Hayes, P. J.
The Logic of Frames.
In D. Metzing (editors), *Frame Conceptions and Text Understanding*, pages 46-61. Walter de Gruyter and Co., Berlin, 1979.
reprinted in *Readings in Knowledge Representation*.

[Hays 73] Hays, D. G.
Types of Processes on Cognitive Networks.
In Zampolli, A. and Calzolari, N (editors), *Computational and Mathematical Linguistics*, pages 323-352. Leo S. Olschki Publishers, 1973.

[Hendrix 79] Hendrix, G. G.
Encoding Knowledge in Partitioned Networks.
In N. V. Findler (editors), *Associative Networks: Representation and Use of Knowledge by Computers*, pages 51-92. Academic Press, New York, 1979.

[Israel and Brachman 81]
Israel, D. J. and Brachman, R. J.
Distinctions and Confusions: A Catalogue Raisonne.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 452-459. Vancouver, BC, 1981.

[Kaczmarek *et al.* 86]
Kaczmarek, T. S., Bates, R. and Gabriel, R.
Recent Developments in NIKL.
In *Proceedings of AAAI-86*, pages 978-985. American Association of Artificial Intelligence, Philadelphia, PA, 1986.

[Kass *et al.* 88] Kass, R., Katriel, R. and Finin, T.
Breaking the Primitive Concept Barrier.
In *The Fourth Conference on Artificial Intelligence Applications*, pages 66-73. IEEE Computer Society, 1988.

[Kastner *et al.* 86] Kastner, J., Apte, C., Griesmer, J., Karnaugh, M., Mays, E. and Tozawa, Y.
A Knowledge Based Consultant for Financial Marketing.
AI Magazine 7(5):71-79, 1986.

[KEE User's Guide 88]
Intellicorp, Inc.
KEE User's Guide.
May, 1988

[Lebowitz 85] Lebowitz, M.
RESEARCHER: An Experimental Intelligent Information System.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 858-862. Los Angeles, CA, 1985.

[Lehnert and Wilks 79]
Lehnert, W. and Wilks, Y.
A Critical Perspective on KRL.
Cognitive Science 3(1):1-28, 1979.

[Lenat *et al.* 86] Lenat, D., Prakash, M. and Shepherd, M.
CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks.
AI Magazine 6(4):65-85, 1986.

[Levesque 86a] Levesque, H. J.
Knowledge Representation and Reasoning.
Annual Review of Computer Science.
Annual Reviews, Inc., Palo Alto, Ca., 1986, pages 35-51.

[Levesque 86b] Levesque, H. J.
Making Believers out of Computers.
Artificial Intelligence 30(1):81-108, 1986.

[Levesque and Brachman 85]
Levesque, H. J. and Brachman, R. J.
A Fundamental Tradeoff in Knowledge Representation and Reasoning.
In R. J. Brachman and H. J. Levesque (editors), *Readings in Knowledge Representation*, pages 42-70. Morgan Kaufmann, Los Altos, CA, 1985.

[Levesque and Brachman 87]
Levesque, H. and Brachman, R. J.
Expressiveness and Tractability in Knowledge Representation and Reasoning.
Computational Intelligence (3):78-93, 1987.

[MacGregor 88] MacGregor, R.
A Deductive Pattern Matcher.
In *Proceedings of AAAI-88*, pages 403-408. American Association of Artificial Intelligence, Saint Paul, MN, 1988.

[MacGregor and Bates 87]
MacGregor, R. and Bates, R.
The LOOM Knowledge Representation Language.
Technical Report ISI/RS-87-188, USC/Information Sciences Institute, Marina del Ray, CA, 1987.

[MacGregor and Brill 89]
MacGregor, R. and Brill, D.
LOOM Reference Manual (Draft).
USC/Information Sciences Institute, Marina del Ray, CA.
April, 1989

[Mark 81] Mark, W.
Representation and Inference in the Consul System.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 375-381. Vancouver, BC, 1981.

[Mays 88] Mays, E.
Object Centered Knowledge Representation.
IBM T. J. Watson Research Center, Yorktown Heights, NY.
October, 1988
Tutorial Presented at ITL Conference.

[Mays *et al.* 91a] Mays, E., Dionne, R. and Weida, R.
K-Rep System Overview.
Sigart Bulletin 2(3):93-97, June, 1991.
Special Issue on Implemented Knowledge Representation and Reasoning Systems.

[Mays *et al.* 91b] Mays, E., Lanka, S., Dionne, B. and Weida, R.
A Persistent Store for Large Shared Knowledge Bases.
IEEE Transactions on Knowledge and Data Engineering 3(1):33-41, March, 1991.
An earlier version appeared in IEEE CAIA-90, pages 169-175.

[McDermott 76] McDermott, D.
Artificial Intelligence Meets Natural Stupidity.
SIGART Newsletter (57), April, 1976.

[McDermott 78] McDermott, D.
Tarskian Semantics, or No Notation Without Denotation!
Cognitive Science 2(3):277-282, 1978.

[Minsky 75] Minsky, M.
A Framework for Representing Knowledge.
In P. Winston (editor), *The Psychology of Computer Vision*, pages 211-277. McGraw Hill, New York, 1975.

[Moore 82] Moore, R. C.
The Role of Logic in Knowledge Representation and Commonsense Reasoning.
In *Proceedings of AAAI-82*, pages 428-433. American Association of Artificial Intelligence, Pittsburgh, PA, 1982.

[Moser 83] Moser, M. G.
An Overview of NIKL, The New Implementation of KL-ONE.
Research in Knowledge Representation for Natural Language Understanding (BBN Report no. 5421).
Bolt Beranek and Newman, Cambridge, MA, 1983, pages 7-26.

[Nebel 88] Nebel, B.
Computational Complexity of Terminological Reasoning in BACK.
Artificial Intelligence 34(3):371-383, 1988.

[Nebel 90a] Nebel, B.
Reasoning and Revision in Hybrid Representation Systems.
Springer Verlag, Berlin; New York, 1990.

[Nebel 90b] Nebel, B.
Terminological Reasoning is Inherently Intractable.
Artificial Intelligence 43:235-249, 1990.

[Newell 82] Newell, A.
The Knowledge Level.
AI Magazine 2(2):1-20, 1982.

[Newell and Simon 72]
Newell, A. and Simon, H. A.
Human Problem Solving.
Prentice Hall, Englewood Cliffs, NJ, 1972.

[Newell and Simon 76]
Newell, A. and Simon, H. A.
Computer Science as Empirical Inquiry: Symbols and Search.
Communications of the ACM 19(3):113-126, March, 1976.

[Nilsson 80] Nilsson, N. J.
Principles of Artificial Intelligence.
Tioga, Palo ALto, CA, 1980.

[Patel-Schneider 84]
Patel-Schneider, P. F.
Small can be Beautiful in Knowledge Representation.
In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pages 11-16. Denver, CO, 1984.

[Patel-Schneider 89]
Patel-Schneider, P. F.
Undecidability of Subsumption in NIKL.
Artificial Intelligence 39(2):263-272, 1989.

[Patel-Schneider *et. al.* 84]
Patel-Schneider, P. F., Levesque H.J., and Brachman, R. J.
ARGON: Knowledge Representation meets Information Retrieval.
In *Proceedings of the First Conference on Artificial Intelligence Applications*. IEEE Computer Society, Denver, CO, 1984.

[Quillian 68] Quillian, M. R.
Semantic Memory.
Semantic Information Processing.
MIT Press, Cambridge, MA, 1968, pages 227-270.

[Reiter 78] Reiter, R.
On Reasoning by Default.
In *Proceedings of Theoretical Issues in Natural Language Processing-2*, pages 210-218. University of Illinois at Urbana-Champaign, 1978.
reprinted in *Readings in Knowledge Representation*.

[Rich, C. 82] Rich, C.
Knowledge Representation Languages and Predicate Calculus: How to Have Your Cake and Eat It Too.
In *Proceedings of AAAI-82*, pages 193-196. American Association of Artificial Intelligence, Pittsburgh, PA, 1982.

[Rich, C. 85] Rich, C.
The Layered Architecture of a System for Reasoning about Programs.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 540-546. Los Angeles, CA, 1985.

[Rich, E. 83] Rich, E.
Artificial Intelligence.
McGraw Hill, New York, 1983.

[Roberts and Goldstein 77]
Roberts, R. B. and Goldstein, I. P.
The FRL Primer.
Technical Report AI Memo No. 408, MIT/Artificial Intelligence Laboratory, Cambridge, MA, 1977.

[Robins 86] Robins, G.
The NIKL Manual.
Technical Report, USC/Information Sciences Institute, Marina del Ray, CA, 1986.

[Schank 72] Schank, R. C.
Conceptual Dependency: A Theory of Natural Language Understanding.
Cognitive Psychology 3:552-631, 1972.

[Schmolze 89a] Schmolze, J. G.
Terminological Knowledge Representation Systems Supporting N-ary Terms.
In *First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 432-443. Toronto, Ontario, Canada, 1989.

[Schmolze 89b] Schmolze, J.
The Language and Semantics of NIKL.
Technical Report 89-4, Tufts University, Medford, MA, 1989.

[Schmolze and Israel 83]
Schmolze, J. and Israel, D.
KL-ONE: Semantics and Classification.
Research in Knowledge Representation for Natural Language Understanding (BBN Report no. 5421).
Bolt Beranek and Newman, Cambridge, MA, 1983, pages 27-39.

[Schmolze and Lipkis 83]
Schmolze, J. G. and Lipkis, T. A.
Classification in the KL-ONE Knowledge Representation System.
In *Proceedings of the International Joint Conference on Artificial Intelligence,* pages 330-332. Karlsruhe, West Germany, 1983.

[Schor *et al.* 86] Schor, M., Daly, T., Lee, H. S. and Tibbits, B.
Advances in RETE Pattern Matching.
In *Proceedings of AAAI-86*, pages 226-232. American Association of Artificial Intelligence, Philadelphia, PA, 1986.

[Schubert 76] Schubert, L. K.
Extending the Expressive Power of Semantic Networks.
Artificial Intelligence 7(2):163-198, 1976.

[Schubert 79] Schubert, L. K.
Problems with Parts.
In *Proceedings of the International Joint Conference on Artificial Intelligence,* pages 778-784. Tokyo, 1979.

[Smith 85] Smith, B. C.
Prologue to "Reflections and Semantics in a Procedural Language".
In R. J. Brachman and H. J. Levesque (editors), *Readings in Knowledge Representation*, pages 32-39. Morgan Kaufmann, Los Altos, CA, 1985.

[Sowa 87] Sowa, J.
Semantic Networks.
Encyclopedia of Artificial Intelligence.
John Wiley and Sons, New York, 1987, pages 1011-1024.

[Steele 84] Steele, G. L. Jr.
Common Lisp: The Language.
Digital Press, Bedford, MA, 1984.

[Szlovits *et al.* 77] Szlovits, P., Hawkinson, L. B., and Martin, W. A.
AN Overview of OWL, A Language for Knowledge Representation.
Technical Report MIT/LCS/TM-86, Laboratory for Computer Science, M.I.T., Cambridge, MA, June, 1977.

[Tanenbaum 87] Tanenbaum, A. S.
Operating Systems: Design and Implementation.
Prentice Hall, Englewood Cliffs, NJ, 1987.

[Tou *et al.* 82] Tou, F., Williams, M., Fikes, R., Henderson, A. and Malone, T.
RABBIT: An intelligent Database Assistant.
In *Proceedings of AAAI-82*, pages 314-318. American Association of Artificial Intelligence, Pittsburgh, PA, 1982.

[Touretzky 84] Touretzky, D. S.
Implicit Ordering of Defaults in Inheritance Systems.
In *Proceedings of AAAI-84*, pages 322-325. American Association of Artificial Intelligence, Austin, Texas, 1984.

[Touretzky 86] Touretzky, D. S.
Research Notes in Artificial Intelligence: *The Mathematics of Inheritance Systems.*
Morgan Kaufmann, Los Altos, CA, 1986.

[Vilain 85] Vilain, M.
The Restricted Language Architecture of a Hybrid Representation System.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 547-551. Los Angeles, CA, 1985.

[Weida 88] Weida, R. A.
K-Rep Summer Project.
IBM T. J. Watson Research Center, Yorktown Heights, NY.
1988
Internal Memorandum.

[Winston 84] Winston, P. H.
Artificial Intelligence, 2nd. ed.
Addison Wesley, Reading, MA, 1984.

[Woods 75] Woods, W. A.
What's in a Link: Foundations for Semantic Networks.
Representation and Understanding: Studies in Cognitive Science.
Academic Press, New York, NY, 1975, pages 35-82.

[Woods 86] Woods, W. A.
Important Issues in Knowledge Representation.
Proceedings of the IEEE 74(10):1322-1334, 1986.

[Yen *et al.* 89] Yen, J., Neches, R. and MacGregor, R.
Classification-based Programming: A Deep Integration of Frames and Rules.
Technical Report ISI/RS-88-213, USC/Information Sciences Institute, Marina del Ray, CA, 1989.

Table of Contents

List of Figures