Computers Math. Applic. Vol. 23, No. 2–5, pp. 133–177, 1992
Printed in Great Britain. All rights Reserved

0097-4943/92 $5.00 + 0.00
Copyright©1992 Pergamon Press plc

THE KL-ONE FAMILY

WILLIAM A. WOODS*
Aiken Computation Laboratory, Harvard University
Cambridge, MA 02138, U.S.A.
JAMES G. SCHMOLZE
Department of Computer Science, Tufts University
Medford, MA 02155, U.S.A.

Abstract — The knowledge representation system KL-ONE has been one of the most influential and imitated knowledge representation systems in the Artificial Intelligence community. Begun at Bolt Beranek and Newman in 1978, KL-ONE pioneered the development of taxonomic representations that can automatically classify and assimilate new concepts based on a criterion of terminological subsumption. This theme generated considerable interest in both the formal community and a large community of potential users. The KL-ONE community has since expanded to include many systems at many institutions and in many different countries. This paper introduces the KL-ONE family and discusses some of the main themes explored by KL-ONE and its successors. We give an overview of current research, describe some of the systems that have been developed, and outline some future research directions.

1. INTRODUCTION

In 1975, "What's in a Link" [1] challenged the semantic network research community to think more clearly and be more explicit about the meanings of links and their uses in such networks. One of the issues raised was a distinction between structural and assertional links—the former link together constituents of a composite conceptual structure, while the latter express assertions about the concepts that they link. Such a distinction is important, for example, in order to know whether a link structure such as:[1]

[telephone] [color] [black]

is intended to represent the description [black telephone] (i.e., the category of telephones that are black) or the assertion that telephones are black (i.e., a sentence expressing the claim that all telephones are black).

Building on this distinction, and partially in answer to the challenges of "What's in a Link," Ron Brachman, in his Harvard Ph.D. thesis, developed a set of conventions for representing structured concepts in "Structured Inheritance Networks" [2]. In these networks, the components of structured concepts were made explicit and concepts could be defined in terms of other concepts. Moreover, the relationships of various parts of structured concepts to corresponding parts of other more general and more specific concepts were also explicitly represented. The resulting network, among other things, effectively organized defined concepts into a partial ordering based on a relationship of defined specialization. The definitions of concepts were thus taken seriously, since it is the definition of a concept that determines its place in the partial ordering.

*We gratefully acknowledge Ron Brachman, Fritz Lehmann, Bob MacGregor, Bill Mark, Peter Patel-Schneider, and Marc Vilain for many helpful comments on earlier drafts of this paper. We also thank all of the people who responded to our request for references and assistance. This research was supported in part by a grant from the Kapor Family Foundation.

[1]Throughout the paper, we will denote concepts with English names or descriptions enclosed in square brackets, e.g., [person].

Brachman's thesis led to the development of the knowledge representation system KL-ONE [3,4] as part of the Natural Language Understanding Project at Bolt Beranek and Newman Inc. (BBN). The first author was the principal investigator for this project, and the second author was a key developer of the system.[2] KL-ONE used structured inheritance networks to construct taxonomic structures that ordered concepts on the basis of generality and (because of KL-ONE's explicit notational semantics) permitted an operation of automatic classification that could assimilate new concepts into the network at the correct positions in the taxonomy.

KL-ONE was used at a number of other institutions, most notably at the Information Sciences Institute of the University of Southern California (USC/ISI) (e.g., CONSUL [5–8] used KL-ONE; and Explainable Expert Systems (EES) [9,10], the Penman natural language generator [11] and parser [12,13], and the Integrated Interfaces Display System [14] all used KL-ONE's successor, NIKL). Moreover, the issues that it raised were actively investigated by researchers from many institutions in a series of specialized workshops, the second of which is summarized in [15]. Many of these issues were embodied and explored in other systems at other institutions, and KL-ONE thus became the root of a family of systems, all pursuing a persistent, if somewhat subtle, collection of research themes that has matured and evolved over time. MacGregor [16] provides a brief overview of the history of KL-ONE and its successors that complements the one presented here.

In this paper, we introduce KL-ONE and some of its successors and give an overview of the research underlying the development of these systems. We begin with an introduction to the basic representational mechanisms of KL-ONE, followed by an exposition of the research themes pursued by KL-ONE and its successors. This will be followed by a discussion of a number of research projects and experimental systems, concluding with a summary of the current state of this research. Finally, we will present some comments regarding directions for future work.

2. THE KL-ONE PERSPECTIVE

A major theme of KL-ONE and its successors is that the semantics of one's representational devices should be well understood. That is, the meanings of represented concepts should be unambiguously determined by explicit notational devices whose meanings (semantics) are understood, so that algorithms can operate on the representation in accordance with the semantics of the notation, without needing ad hoc provisions for specific domain concepts. Representations with this characteristic have come to be known as "principled" knowledge representations. Without a principled semantics, there is little that can be done in a general and extensible way and there is no way a system can automatically classify and handle new concepts that were not programmed into it. Moreover, it is difficult for an unprincipled system to evolve gracefully, because eventually it becomes too difficult to determine the ramifications of adding a new piece of knowledge when the existing structure embodies arbitrary relationships with no governing principles. KL-ONE began a tradition of developing principled notations for expressing conceptually distinct meanings of the kind found in natural language statements. Subsequent systems have gone further in this direction by providing an explicit model-theoretic semantics (see Section 4.4.).

Initially, the KL-ONE project set out to develop a set of representational conventions that would be sufficient to express any concept expressible in natural language. Brachman, in his thesis, called these conventions "epistemological" primitives. They might have been better termed "concept structuring" primitives, since they were not dealing with sources of knowledge or justifications for belief, but were essentially primitives of abstract representational notation, as contrasted with primitive domain concepts. That is, these primitives dealt with basic conceptual relationships such as a concept having an attribute, satisfying a constraint, being defined by a set of properties, being more specific than another concept, etc. Concept structuring primitives are contrasted with "primitive" domain concepts such as "ship," "tank," "bagel," "transistor," or whatever the subject matter of the knowledge base might be. Concept structuring primitives

[2]Many other people participated in the project, including, among others: Madeline Bates, Rusty Bobrow, Ron Brachman, Jeff Gibbons, Brad Goodman, David Israel, Hector Levesque, Tom Lipkis, Bill Mark, Candy Sidner, Bill Swartout, Dave Wilczynski, Marc Vilain, Martin Yonke, and Frank Zydbel.

(perhaps together with some logical primitives for things like sets and sequences) should be the only primitives on which the reasoning algorithms of a system depend.

2.1. Comparison with Typical Frame Systems

In order to understand the KL-ONE world view, it is useful to contrast it with the world view of a typical frame-based representation system. Typically, a frame system consists of a collection of data structures called "frames" that can be thought of as standing for classes of objects that have attributes. Each frame has a number of data elements called slots, each of which corresponds to an attribute that members of its class can have. Each slot contains information about the corresponding attribute, such as default values, restrictions on possible fillers, attached procedures or methods for computing (nondefault) values when needed, and procedures for propagating side effects when the slot is filled. A frame is essentially a data structure for organizing certain kinds of computation.

Typically a frame will include an "isa" or "ako" pointer to a more general frame or frames from which additional slots with default values and other information may be inherited. Intuitively these labels correspond to asserting that the first concept is an instance of the second ("isa" = "is a") or that the first concept is a subkind of the second ("ako" = "a kind of"). Early frame systems did not make a distinction between subkinds (e.g., a dog is a mammal) and instances (e.g., Fido is a dog), but many now do, partly in response to papers such as [1] and [17].

In a typical frame system, these "isa" and "ako" pointers are entered by the person who constructs the data base, and their (operational) semantics is defined by the inheritance mechanism of the system. Often this inheritance is defined by the notion of a virtual copy [18], in which a frame is thought of as having its own private copy of all inherited slots and default values. This virtual copy can then be changed locally in order to override inherited defaults. Thus, the semantics of such links is strictly operational—defined by how they work and what they cause to happen. There is no formal criterion for when such links should be added to a frame. It is simply up to the person entering the information to decide where a concept should be inserted into the hierarchy and what its links should be. There is no external criterion of correctness to which these decisions should adhere.

From the KL-ONE perspective, however, the semantics of "ako" relationships should be defined by an external semantic criterion, independent of the data structure and the algorithms that will operate on it. This criterion expresses what it means for one concept to be in an "ako" relationship to another and determines when it is appropriate for the data structure to include such links. Thus, in KL-ONE, such links have a "criterial" as opposed to an "operational" semantics. There is an external criterion for the correctness for a link, determined by what the link "means." This criterion is independent of the subsequent processing that the link will cause or enable. The operational consequences are then justified in terms of the criterial semantics.[3] A criterial semantics can be provided in a number of ways, including combinations of English characterizations of intended meanings, reductions to well understood formal systems such as the first-order predicate calculus, and/or direct model-theoretic semantic accounts. What's important is that there needs to be an explicit understanding of what the notational devices of the system mean that is clearly understood and shared by the implementers of the system and the people who will use it. It is also important that these notational devices be sufficiently expressive to permit the users to correctly say what they want to say within the delineated semantics.

If a system has a criterial semantics, then a person entering information need be concerned only with expressing facts correctly (according to those semantics) without having to anticipate all of the operational consequences. If the implementation is faithful to its criterial semantics, then one can be confident that the resulting behavior will be correct. Further, if knowledge has been entered faithfully according to a criterial semantics, then operational aspects of a system can be changed and extended in a manner consistent with those semantics without having to revise the knowledge already recorded. This division of labor can greatly facilitate the development of complex knowledge-based systems. In fact, one can argue that for developing and maintaining

[3]See [19] for a general discussion of the distinction between criterial and operational semantics.

large knowledge bases, which must necessarily be constructed incrementally and evolved over time, a dependably "criterial" knowledge base manager is essential.

KL-ONE can be thought of as a kind of frame-based system with an additional layer of representational conventions that carry a criterial semantics. These semantics make it possible to perform certain inferential operations such as automatic classification of new concepts. KL-ONE also has a few additional structures not present in most frame systems. The next few sections will describe these more fully.

2.2. Automatic Classification

In virtually every semantic network or frame based system, there is at least one link or slot, such as the above "ako" and "isa," that relates more specific concepts to more general concepts. Other names for such links include: "kind of," "subset of," "member of," "subconcept of," "subkind of," "superconcept," "superc," etc. These links are used to organize concepts into a partially ordered structure called a "taxonomy." The taxonomy is used to record information at appropriate levels of generality thus making it available to more specific concepts by means of inheritance. Classical taxonomies, such as those in biology, are usually strict hierarchies (in which each class has a unique parent class), but in KL-ONE, as in many knowledge representation systems, concepts can have multiple parents.

If a representational notation is sufficiently well defined by a criterial semantics and structured concepts can be defined in terms of other concepts, then many generality relationships among composite, structured concepts can be derived automatically from their structures. For example, the concept "a woman with children" can be inferred to be more general than "a woman with sons" by virtue of the structures of the two concepts and the relationship between children and sons. The more general concept in such a case is said to "subsume" the more specific one. The fact that KL-ONE allows one to construct defined concepts and that its notational semantics permits the detection of subsumption relationships between defined concepts makes it possible to automatically assimilate new concepts into a taxonomy by "classifying" them with respect to the taxonomy and adding appropriate ako relationships. "Automatic classification" refers to the ability to insert a new concept into a taxonomy so that it is directly linked to the most specific concepts that subsume it (i.e., are more general than it is) and to the most general concepts that it in turn subsumes.

The ability to automatically classify structured concepts with respect to a taxonomy is one of the distinguishing characteristics of KL-ONE and its successors. Prior to KL-ONE, network taxonomies were completely hand-crafted, with each concept placed in the taxonomy by a human designer. The ability to automatically classify structured concepts has emerged as an important issue in knowledge representation [20].

The concept of automatic classification originated in a pair of algorithms conceived by the first author as important algorithms to be supported by a knowledge representation system [21]. The first algorithm (called MSS, for "most specific subsumer") locates the most specific concepts in a taxonomy that subsume a given input description. The second algorithm (called MGS, for "most general subsumee") locates the most general concepts that are subsumed by an input description. Automatic classification consists of applying these two algorithms and linking the new concept to its most specific subsumers and its most general subsumees. A parallel, marker-passing algorithm for MSS was designed by the first author [21], who also implemented a sequential version as part of an English-like interface to KL-ONE, called JARGON [22]. The JARGON MSS algorithm served as the first KL-ONE classifier used by the BBN natural language understanding system [23]. A substantially expanded classifier was subsequently implemented by Tom Lipkis [24–26].

Automatic classification, based on subsumption of structured concepts, has been one of the major innovations in knowledge representation research and one of the powerful driving themes underlying the considerable interest in KL-ONE. An automatic classification algorithm could not be implemented in a standard frame system without imposing an additional layer of representational convention that distinguishes certain slots and slot attributes as constituting the defining properties of a structured concept. Not all slots and slot attributes in a frame system can be interpreted in this way (in fact most are not). For example, slot attributes that record default

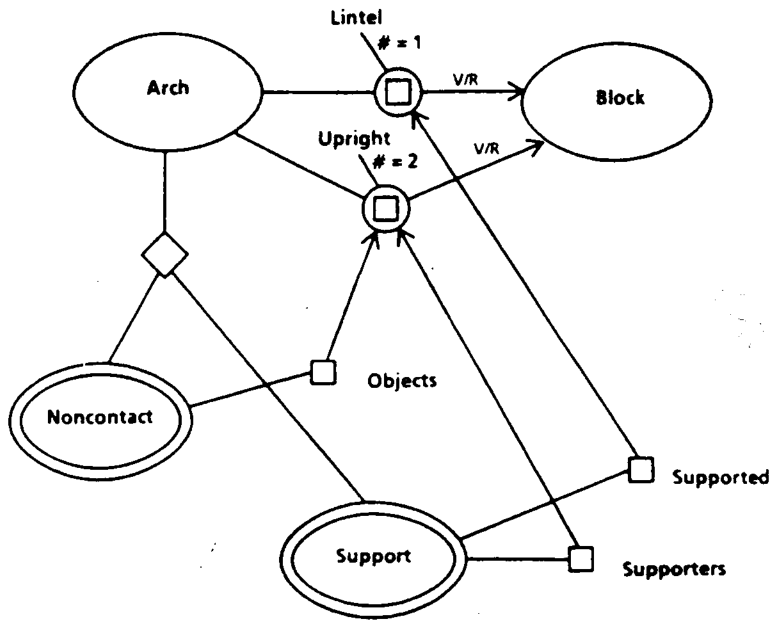

Figure 1. A KL-ONE diagram of a simple "Blocks-World" arch.

values and attached procedures are not defining properties and do not participate as criteria in the subsumption relationship.[4] At a minimum, in order for an automatic classification algorithm to be possible, one would have to distinguish the slots corresponding to defining attributes and specify the intended semantics of those slots. Moreover, one would have to distinguish concepts (frames) that are completely defined by their attributes from those that are merely partially described. If such distinctions are made in the representation and are followed when recording information, then a classification algorithm becomes possible. Some further work is required to address the effect of classification on slots and slot attributes that do not participate in the classification. [28] gives one the framework for doing this.

In summary, automatic classification is made possible by a layer of representational conventions with explicit criterial semantics imposed on top of implementational data structures equivalent to those of a typical frame system. Such conventions dictate that default values and procedures attached to a slot play no part in determining subsumption relationships. KL-ONE's representational conventions (i.e., its concept structuring primitives) constitute such a representational layer which is above the layer of implementational structure and below the layer of conceptual knowledge of domain specific concepts.[5] From the KL-ONE perspective, the important issues are these representational conventions and their consequences—not the implementational data structures themselves. Indeed, the designers of KL-ONE (including the authors) sought to formalize the abstract structure of concepts, independent of particular implementational mechanisms. An important belief of many researchers in the KL-ONE tradition is that the represented knowledge is about something in the world, not merely a data structure.

2.3. *An Introduction to KL-ONE*

KL-ONE formalized the notion of a structured concept as a constellation of elements standing in specified relationships to each other. The relationships of these elements to the concept as a whole are called "roles," and the relationships that the fillers of these roles must have to each other are called "structural conditions." Roles in KL-ONE were intended to capture a common

[4]Brachman [27] pointed out that a frame system that permits any slot attribute to be overridden (i.e., treats all slot attributes as defaults) is inconsistent with automatic classification.

[5]The notion of distinct conceptual layers of representation is becoming recognized in cognitive and computer science as people have begun to realize that the engineering practice of layered architectures is more than just a notational convenience. This point has been argued by Brachman's explicit introduction of the so-called "epistemological" level [29] and Newell's theory of the knowledge level [30].

generalization of the notions of attribute, part, constituent, feature, parameter, functional argument, grammatical case, etc. For example, consider the classical "blocks-world" "arch" (the prototypical structured object in artificial intelligence research), which consists of a rectangular block (called the "lintel") supported horizontally on top of two vertical blocks (the uprights). This concept would have roles for the lintel and the two uprights and would have structural conditions requiring that the lintel be supported by the uprights and that the uprights have open space between them. This is illustrated in Figure 1. Here, concepts are represented by ovals, roles are represented by circled squares, and structural conditions are represented by doubled ovals attached to a diamond shaped lozenge. More details of the graphical notation will be explained below.

In a similar fashion, the concept of a product of two numbers would have roles for multiplier, multiplicand and result. Likewise, an English sentence would have roles for subject, verb, object, etc., and its underlying interpretation might have "case" roles for agent, beneficiary, manner, means, etc.

KL-ONE Notations

KL-ONE has used a number of notations for its structures—some graphical and some lexical. These notations, as well as various internal implementation structures, are all thought of simply as different manifestations of a common abstract conceptual structure. This abstract conceptual structure has properties and consequences that are independent of particular data structures and algorithms, and these properties define the criteria by which inferential algorithms are judged valid.

In KL-ONE's graphical notation, concepts are represented by ellipses and roles are represented by small circles containing inscribed squares. Attached to roles are value restrictions and other role "facets" (e.g., names and number restrictions), and attached to concepts are roles and structural conditions. Value restrictions on roles are concepts that characterize constraints on possible role fillers. They are indicated by directed arrows pointing from the role to a concept that constrains possible fillers. Structural conditions characterize relationships that the fillers of different roles must have to each other. They are indicated by diamond shaped lozenges connecting to structures that encode propositional constraints. These constraints are expressed in special graphical notations equivalent to restricted forms of quantificational logic. These conventions are illustrated in Figure 1 mentioned above. More details of the representation of concepts, roles, and structural conditions will be given shortly.

In addition to the structure within a concept, KL-ONE uses directed arrows to express relationships between concepts and also between roles of concepts. Specifically, directed arrows connect concepts to more general concepts from which they are defined or which they specialize. Directed arrows are also used to connect roles to more general roles that they specialize. The various uses of directed arrows are generally distinguished by the kinds of nodes at the ends of the arrows: concept → concept represents concept specialization, role → role represents role specialization, and role → concept represents value restrictions on the role. Occasionally these arrows will be explicitly labeled (as in the case of role differentiation, to be discussed shortly).

In the semantics of KL-ONE, roles and structural conditions attached to concepts apply to all specializations of those concepts as well. More specific concepts are thus said to "inherit" roles and structural conditions from more general concepts to which they are linked. This allows one to introduce roles and structural conditions into a network at the most general level at which they apply, after which they need not be explicitly repeated for more specific concepts. This form of inheritance is slightly different from that of most frame systems, since what is being inherited are the defining characteristics of a concept, which can be further restricted for more specific concepts, but not overridden.

The specific graphical conventions of the pictorial notation are not important, other than for capturing the relationships among the various parts of a structured concept and making these relationships explicit. However, when KL-ONE was introduced, the use of a common set of explicit graphical conventions, with relatively well-defined semantics, was quickly found to promote a degree of clarity that was not usually present in previous network representations.

KL-ONE structures were found to be sufficiently expressive that a person could read a KL-ONE diagram in terms of the explicit semantics of its representational conventions without having to guess purely from the names of the concepts what the network author had in mind. That is, unlike typical frame based systems, KL-ONE does not need to have concepts with complex names like "animal-that-eats-meat" whose intended meaning is suggested in English but not known to the system or enforced by it.[6] In KL-ONE, such concepts can be explicitly defined in terms of other concepts and their meanings made explicit using the graphical conventions, without needing to appeal to external documentation. Such defined concepts in KL-ONE need not have explicit names in order to be interpretable. (Of course such concepts are defined relative to other concepts, which may be defined in terms of still other concepts, which must eventually be grounded in concepts whose names are recognizable and will mean to the reader what the knowledge base author is intending. These concepts, however, can be common generic concepts with conventionally accepted meanings.)

It should be pointed out that, although the graphical notation is intuitively clear as a pedagogical device for illustrating concepts with small examples, a similar picture containing all of the details in a nontrivial knowledge base would quickly become an unmanageable tangle of lines. The major advantage of network representations for full-scale knowledge bases is not readability but rather the utility that results when algorithms can exploit the links for operational processing. The links are also useful for associative access by network browsing programs that enable a human editor to inspect and modify a knowledge base. It is important to understand that the primary goals of KL-ONE and its successors are to facilitate the construction and use of knowledge bases, not merely the representation of individual sentences.

It should also be pointed out that, despite the goals of KL-ONE for developing well-understood semantics, the original graphical notation is conducive to certain intuitive uses whose semantics are not totally clear. Consequently, work on the formal semantics of KL-ONE and KL-ONE-like systems has tended to use lexical representations that are less intuitive but more amenable to formal analysis (as we do in Section 3). This is progress in some dimensions at the sacrifice of others.

The formal machinery in KL-ONE and subsequent related systems has varied in detail and in spirit from system to system and from researcher to researcher, and has evolved over time as well. For a more complete exposition of the original KL-ONE formalism and its graphic notation, see [4,23]. In the next section, we will describe the key concepts of KL-ONE in a way that is relatively faithful to the original, but using a minimum of notation. In subsequent sections we will introduce a notation in which we can more easily compare the features of subsequent systems.

The Structure of Concepts

A concept in KL-ONE is generally defined by one or more "superconcept" pointers to more general concepts plus a collection of locally attached role descriptions and perhaps some attached "structural conditions." The superconcept pointers specify a class (or classes) of which the defined concept is a subclass, while the role descriptions and structural conditions describe how the concept being defined differs from (the intersection of) its "parent" concepts. In general, concepts can be defined by restricting one or more parent classes by the addition of one or more roles or structural conditions that are not implied by the parents, and/or by tightening the conditions in one or more roles or structural conditions inherited from a parent. A concept can also be defined with no additional roles or structural conditions by merely pointing to more than one superconcept, effectively specifying the conjunction of those parent concepts. For example, the concept [appreciable debt obligation] could be defined by a KL-ONE concept with a superconcept link to [debt obligation] and another superconcept link to [appreciable asset], as in Figure 2. This

[6]There is nothing in a typical frame system that obligates the system to operate on a frame in a way that is consistent with such a name. Consequently, since the name is irrelevant to the system's operation, the author of a knowledge base in such a system does not hesitate to use abbreviated and potentially ambiguous names. As a consequence, many such networks are impossible to read without a prior understanding of how each of the frames is intended to be used.

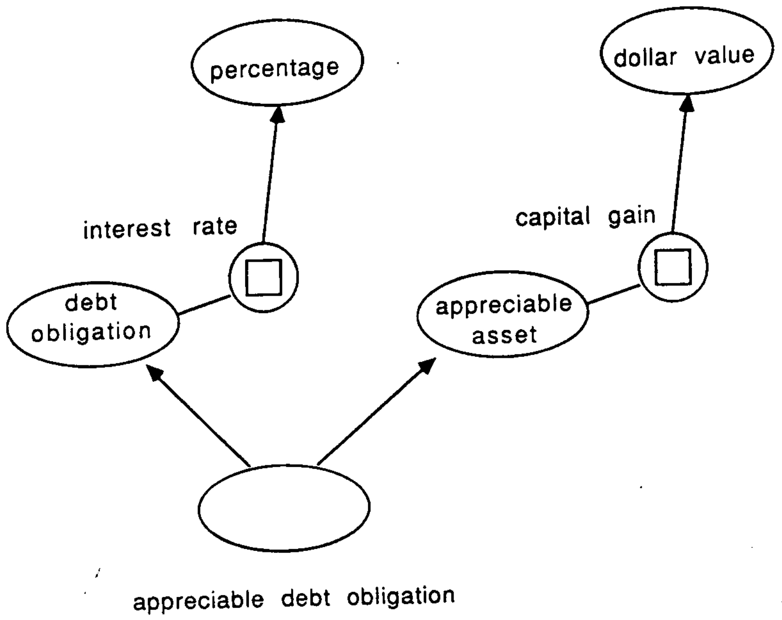

Figure 2. Appreciable debt obligation, an example of conceptual conjunction.

concept would cover investment vehicles such as bonds which both bear an interest rate (like a savings account) and also may incur a capital gain or loss (like a stock). This example illustrates the utility of inheritance from multiple parents, since each of the parent concepts has a well defined use, but instruments such as bonds turn out to have both characteristics and are not totally characterizable in a strict hierarchy as either purely a debt obligation or an appreciable asset. By categorizing bonds under such a conjoined concept, however, one automatically inherits both aspects without having to make a separate copy of the appropriate role descriptions. Note that the name of the defined concept [appreciable debt obligation] is redundant, since one can read the meaning (relative to the two concepts from which it is defined) from its structure.

Primitive Concepts

The above section describes the general case of a defined concept in KL-ONE. However, KL-ONE also permits concepts for which only partial definitions can be given which constrain, but do not fully specify the concept. Such concepts are called "primitive," to indicate that their meanings are not fully defined by the information recorded in the network. Such concepts are common in natural language, especially for what philosophers call "natural kind" terms such as "dog." The claim is that no complete definition of such concepts is possible since one can continually propose borderline cases (wolves or hyenas for example) or exceptions to proposed definitions (hairless dogs for example). For example, a typical dictionary "definition" of "dog" is "a carnivorous domestic animal," which is not a complete definition (since it does not distinguish dogs from cats, for example). An equivalent definition of "dog" in KL-ONE would be represented by a primitive concept to indicate that the defining characteristics are incomplete.

For a primitive concept, the collection of superconcepts, roles, and structural conditions constitute necessary but not sufficient conditions for determining instances of the concept. For nonprimitive concepts, these conditions are both necessary and sufficient. Automatic classification in KL-ONE cannot place a new concept underneath a primitive concept, because it does not know what additional information is required for something to be an instance—i.e., it does not have sufficient conditions. Primitive concepts are indicated in KL-ONE diagrams with an asterisk to indicate that their definition is incomplete.

Note that definitions in KL-ONE are taken seriously. In order for the classifier to work correctly, the representation must distinguish what is fully defined and what is only partly defined. Moreover, unlike in most frame systems, none of these defining properties can be treated as

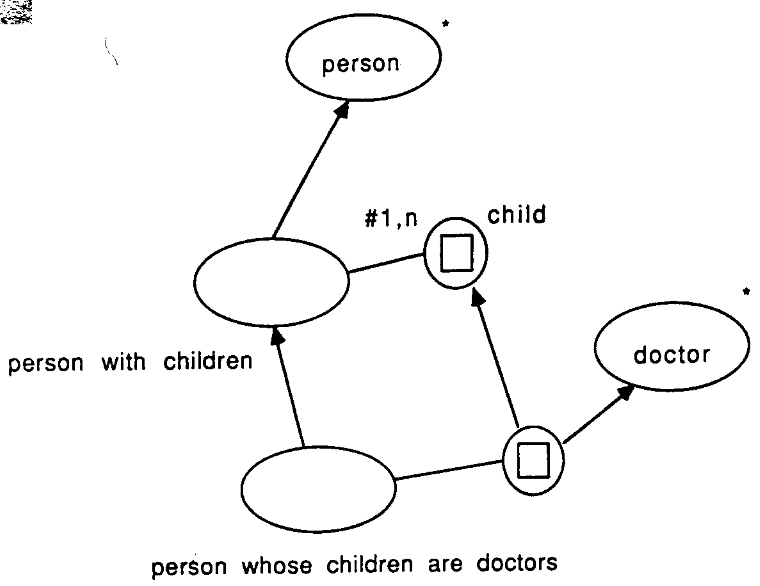

Figure 3. A person whose children are doctors, an example of role restriction.

defaults. In order for the classifier to work, everything in a KL-ONE concept definition must be strict—no cancellation is possible.[7]

The Structure of Roles

Roles in KL-ONE carry information about the kinds of fillers permitted (called "value restrictions"), the minimum and maximum number of allowed fillers (called "number restrictions"), and one or more names. Roles can also be linked to more general roles in one of several ways. First, a role can be a differentiated version of a more general role, e.g., arms and legs are differentiations of the more general role "limb." Second, roles attached to concepts can be modifications or "restrictions" of roles attached to other concepts (or to the same concept). Such restrictions can be obtained by tightening a value restriction and/or a number restriction of a role—e.g., a child role with a value restriction of [doctor] could be a restriction of a child role with a value restriction of [professional]. The effective value restriction of a role restriction is the "intersection" of all of the value restrictions that it has or inherits from parents—in the above case, the intersection of [doctor] and [professional] (which would be [doctor], assuming the network "knows" that doctors are professionals by means of an appropriate superconcept relationship between the two concepts.) The restriction relationship between roles is indicated pictorially by an arrow from the more specific role to the role that it restricts (sometimes labeled "mod" in KL-ONE figures). The differentiation relationship is indicated by a similar arrow labeled "diff."

Roles and role restrictions can be used to define a concept. For example, a concept equivalent to "person whose children are doctors" could be defined as a specialization of [person with children] by using a restriction of the [child] role that adds the value restriction [doctor]. This is illustrated in Figure 3. This figure begins with the primitive concept [person] at the top (note the asterisk indicating that it is primitive). This is used to define the concept [person with children] by the addition of a child role with the number restriction $\#1, n$—indicating at least one and at most n children (where n signifies an arbitrary upper bound). This concept and this role are then used to define [person whose children are doctors] by attaching a role restriction requiring the child role to have a value restriction [doctor]—indicating that its fillers must be doctors. This role restriction inherits the name and number restriction from its parent role since it does not mention either on its own. Note that the names of the two defined concepts are redundant, since the meaning of the concepts, in terms of the primitive concepts [person] and [doctor], are formally specified by the notation.

[7]But see [28] for a discussion of how to combine defining and default properties in KL-ONE-like systems.

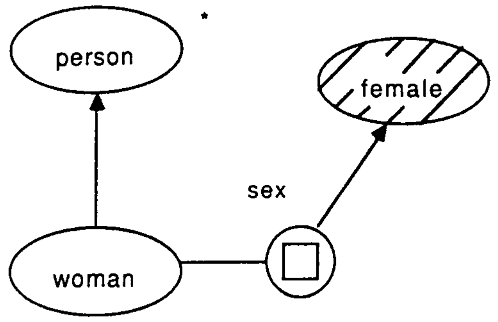

Figure 4. Woman, a person whose sex is female.

You may notice that the topmost [child] role in this figure does not specify any value restriction for its fillers. To be more complete, this role would have a pointer to the concept [person] to indicate that the children of persons are persons. In general, when a role has no value restriction, this is considered to be equivalent to having a value restriction of [THING], the most general possible concept.

In the earliest versions of KL-ONE, concepts and roles were thought of as distinctly different things. In a slightly later version it was realized that roles could be thought of as concepts in their own right, and that role differentiation is analogous to the superconcept relationship between concepts. In subsequent KL-ONE-like systems, concepts have been identified with classes or unary predicates, while roles have been identified with binary relations.

Individual Concepts

KL-ONE makes a distinction between generic and individual concepts, corresponding roughly to the difference between sets and their members. While the superconcept link relates generic concepts at different levels of generality, an "individuates" link is used to link concepts representing individuals to generic concepts of which the individuals are instances. Individual concepts are a distinct type of concept, indicated graphically by diagonal shading within the concept's oval representation. Similarly, the fillers of roles for individual concepts are represented by special individual role instantiations (called "iroles") represented by small squares with diagonal shading inside. The set of fillers of a role (whose individual members may be unknown) is also representable by an explicit graphical convention—a circled square with shading between the circle and the square.

In defining a concept, it is sometimes desirable to specify a role constraint in terms of a specific individual filler rather than a generic concept specifying a constraint on the filler. For example, in defining a woman as "a person whose sex is female," it is important to say that the sex role of a woman is filled by an individual gender called [female]—not by some instance of a generic concept called [female]. A generic concept called [female] should—if the representation is not to mislead readers—denote the class of all female animals and plants, in which case an instance of that concept would be an individual animal or plant, not an individual gender. An appropriate representation of this definition in KL-ONE is indicated in Figure 4.

The distinction between generic and individual concepts is important and has been made in various ways in different knowledge representation formalisms. However, the notion of individual concept embedded in KL-ONE seems to have been confusing and difficult to use in practice. In KL-ONE's successors, individual concepts have been either omitted or dealt with differently, e.g., by having individuals in the assertional component.

Structural Conditions

Structural conditions (also called "structural descriptions") in KL-ONE express required relationships among roles—for example, the constraint that the uprights of a blocks-world arch must support the lintel. Generally, structural conditions are expressed by means of "parametric individuals," descriptions of individual relationships that must be satisfied by instances of the

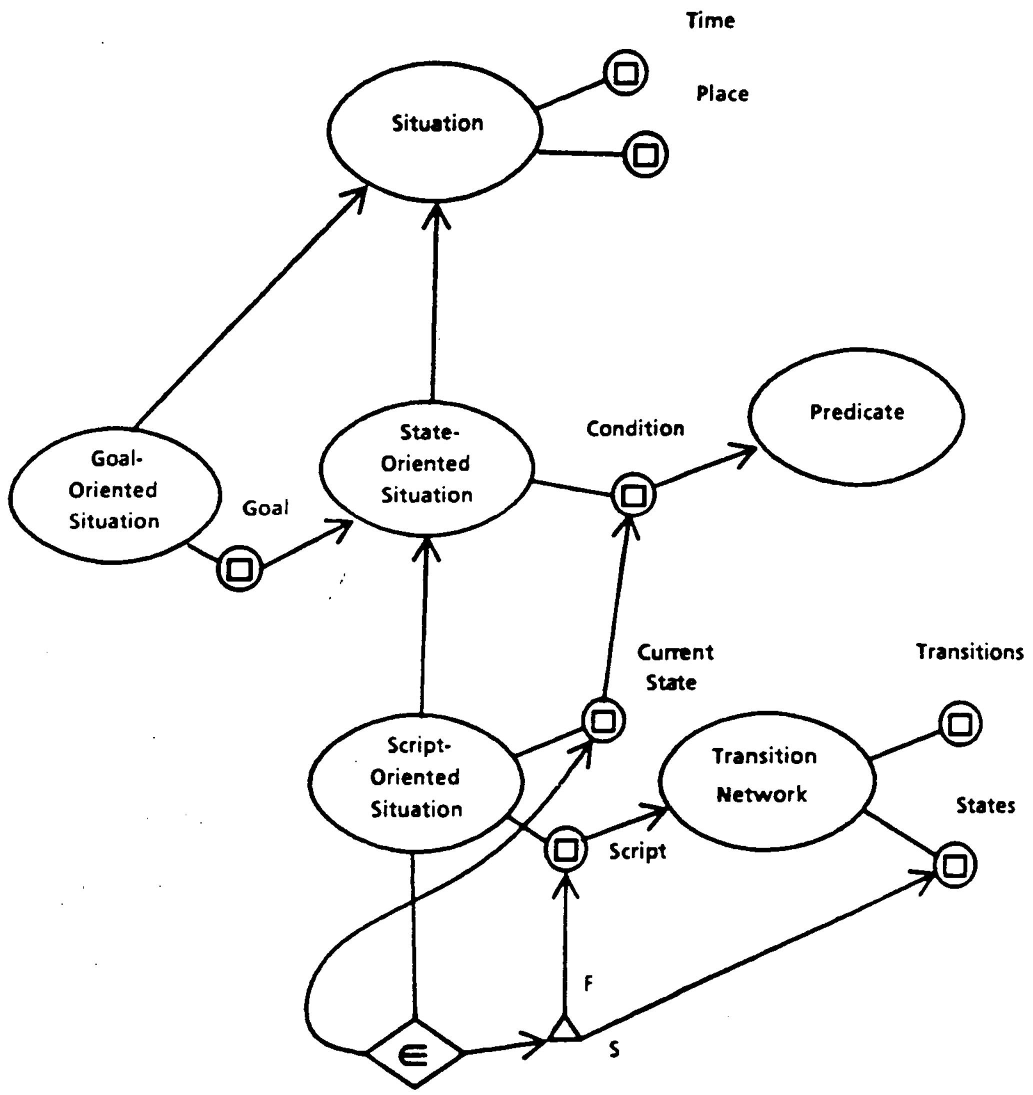

Figure 5. Kinds of situations.

defined concept, such as the individual support relationship that holds between the uprights and the lintel of an arch (see Figure 1). These relationships are called "parametric individuals" because there is a different instantiation of the relationship for each instance of the concept being defined—e.g, a different support relationship for each individual arch. Thus, the support requirement in the arch definition is effectively parameterized by the context of each individual arch instance and stands for a different individual support relationship for each individual arch.

Structural conditions are indicated by doubled ovals attached to a diamond shaped lozenge attached to the concept that they modify. Each such condition refers to a generic concept that is defined elsewhere in the network and specifies an alignment of the roles of that concept with appropriate fillers in the context of the concept being defined. The fillers are generally specified by roles of the concept being defined, or by "role chains" (sequences of roles that serve as an access path from the concept being defined to the intended fillers). For example, in Figure 1, the fillers of the [Supporters] and the [Supported] of the [Support] relationship are specified to be the uprights and the lintel, respectively. Some fillers could also be specified by individual concepts (i.e., constants).

KL-ONE also provides a special kind of structural condition called "role value maps" for the special case of equality and subset relations between the filler sets of roles or role chains. This special case can be expressed using parametric individuals, but can be handled more efficiently as a special case. Role value maps are indicated by including the intended subset relation (either "subset" or "equality") within the diamond lozenge of a structural condition. Role chains are represented using chains of "focus/subfocus" nodes represented by small triangles. These conventions are illustrated in Figure 5, which illustrates a nontrivial use of KL-ONE notation. The figure illustrates a subclassification of situations into goal-oriented and state-oriented situations. The former has a goal that is constrained to be a state-oriented situation, while the latter has an associated condition that is a predicate. A special case of a state-oriented situation is a script-oriented situation, whose condition consists of a current state predicate. Script-oriented situations also have a role called a script, which is filled by a transition network. A transition network has roles for states and transitions. Script-oriented situations have a structural condition (specifically, in this case, a role value map) which asserts that the current state of the script-oriented situation must be one of the states of the transition network that fills its script role. This latter is specified by a role chain leading from the script role of [script-oriented situation] to the state role of [transition network] (via the triangular "focus/subfocus" node that links the elements of the role chain). The role chain is necessary since the constraint does not relate to the states of just any transition network, but only the one that fills the script role of the script-oriented situation in question. This figure shows a set membership symbol in the role value map, indicating that the filler of the current state role is a member of the indicated set of states. Technically, KL-ONE implemented only subset and equality relationships in role value maps. The notation in this figure is effectively a graphical abbreviation expressing the constraints that the condition role has a number restriction of exactly one and that the corresponding singleton roleset is a subset of the indicated set of states.[8]

Nexuses and Contexts

The first implementation of KL-ONE represented assertional information by connecting concepts in the conceptual taxonomy to objects called "nexuses" in hypothetical contexts. Ordinarily, nexuses represented distinct individuals in the context and an individual nexus could be connected to all of the different conceptual descriptions that it satisfied. In addition, special kinds of nexuses could represent "individuals" that might or might not exist and might or might not be identical with other individuals. For example, the unknown murderer in a murder mystery could be represented by a separate, nonunique nexus without prejudging his or her distinctness from other individuals in the context. A nonunique nexus is treated as a hypothetical individual with properties, but is not counted as a distinct individual when the total number of individuals are counted or when distinctness judgements are being made. Some of the KL-ONE nexus mechanisms were attempts to explore representations of intensional structures not expressible in first-order logic and are of considerable importance. However, they were never fully worked out, although a limited version of the mechanism was implemented in KL-ONE. Such devices have not been pursued by KL-ONE's successors.

2.4. Taxonomic Structure

Concepts in KL-ONE are linked to more general concepts by the superconcept relation. The more general concept in such a relationship is called the superconcept and is said to subsume the more specific subconcept. The superconcept relationship effectively organizes the concepts in KL-ONE into a taxonomy on the basis of generality, and relationships of differentiation and restriction between roles indicate correspondences between different parts of related concepts. Some of the uses of such structures begin to become apparent in the figure we have just seen (Figure 5). Such structures can be useful to a person simply for clarifying one's conceptions of a problem domain. For a computer, they permit a variety of special types of reasoning, especially classification and inheritance.

The taxonomic structure in KL-ONE is used to organize structured concepts at different levels of generality. The classification operation is used to place new descriptions into a taxonomy at their correct positions. The taxonomy in KL-ONE has been used for a variety of purposes. In

[8]Many other details are omitted from this figure for expository purposes—for example, no number restrictions are shown, several roles have no value restrictions, and only one structural condition is shown.

the application called AIPS [31–33], the taxonomy was used as a conceptually-oriented, object-oriented programming system to organize display instructions and graphic presentation algorithms for an advanced information presentation system. In a system called IRUS, the taxonomy was used to organize the semantic interpretation process for a natural language understanding system [34]. In CONSUL [5–8], it was used to organize information relating a user's world model to the command structure of a computer message system so that a user's description of what he or she wanted to do could be transformed into the appropriate commands for doing it.

The kind of taxonomic structure that KL-ONE is intended to support is illustrated by the example in Figure 6. The figure represents the information used by the AIPS system discussed above to generate graphical displays of ATN grammars, using KL-ONE as an object-oriented system for graphical presentation. ATN grammars [35] are a grammar formalism based on transition network diagrams that lend themselves to graphical presentation and editing and have become a standard engineering technique for expressing complex grammars for natural languages in a form that can be efficiently processed by computer. In the next section, we will look at this taxonomy in some detail.

An Example

Figure 6 shows a KL-ONE taxonomy that was taken from an application using AIPS and IRUS to display portions of an ATN grammar in response to English commands [23]. To understand the example, it is useful to know that ATN grammars are a generalization of nondeterministic pushdown store automata and that they represent possible sequences of constituents as transitions in a state transition network with a specified start state and a set of distinguished final states. Each transition in the diagram is labeled with the kind of constituent that can enable the transition, together with additional conditions required for the transition and actions to be taken as a consequence of the transition. There are a number of different types of transitions, expressed as different types of "arcs" connecting states in the grammar. For example, CAT and WRD arcs pick up individual words from the input string, JUMP arcs change state without consuming anything from the input string, and POP arcs indicate the completion of a constituent.

The information shown in Figure 6 was developed in the context of the AIPS system [31–33] and the Natural Language Understanding Project [36] at BBN. It illustrates how specific domain knowledge (in this case knowledge of the structure of ATN grammars) can be conveniently integrated with general purpose graphic presentation capabilities to produce displays with a minimum of effort. In this case, all that was necessary was to describe the various types of ATN grammar structures and to specify at a high level of abstraction that states are to be displayed as circles containing the state name, and that arcs are to be displayed as three segment arrows with an arc label on a horizontal middle segment.

At the top of the figure is the general concept of an ATN constituent, which is subdivided immediately below into states and arcs. An ATN constituent has a role called "display form" which is constrained to be an ATN display form. ATN display forms are subdivided into state display forms and arc display forms (defined elsewhere in the network and not shown here, but specifying the graphical display conventions described above). The high level display form role of ATN constituents is modified at the level of states and arcs to have values that are state display forms and arc display forms, respectively. This information constitutes the entire interface between the domain model of ATN grammars and the graphical display routines. The remainder of the figure specifies information about ATN grammars.

The concept [STATE] has a role named "arc," whose value restriction is the concept [ARC]. The concept [ARC] has roles for the tests and actions that an ATN arc can perform and a "source state" arc whose value restriction is [STATE]. Arcs are subdivided into pop arcs and connecting arcs, where only the latter have "next states" (which again are states). Connecting arcs are in turn subdivided into jump arcs and consuming arcs, where only the latter have a label for what is consumed. Consuming arcs are in turn subdivided into input-consuming arcs and "vir" (for "virtual") arcs, where the former consume a constituent from the input string while the latter consume an element from a special "hold" list of constituents found previously in the sentence. In a similar fashion, the figure shows several different kinds of input-consuming arcs: CAT arcs, WRD arcs, and TST arcs. Note that, although not explicitly so marked in the figure, all of these concepts are primitive concepts in KL-ONE terminology, since what is shown are necessary characteristics but not sufficient conditions. For example, there is not enough information in the diagram to distinguish CAT arcs from WRD arcs.

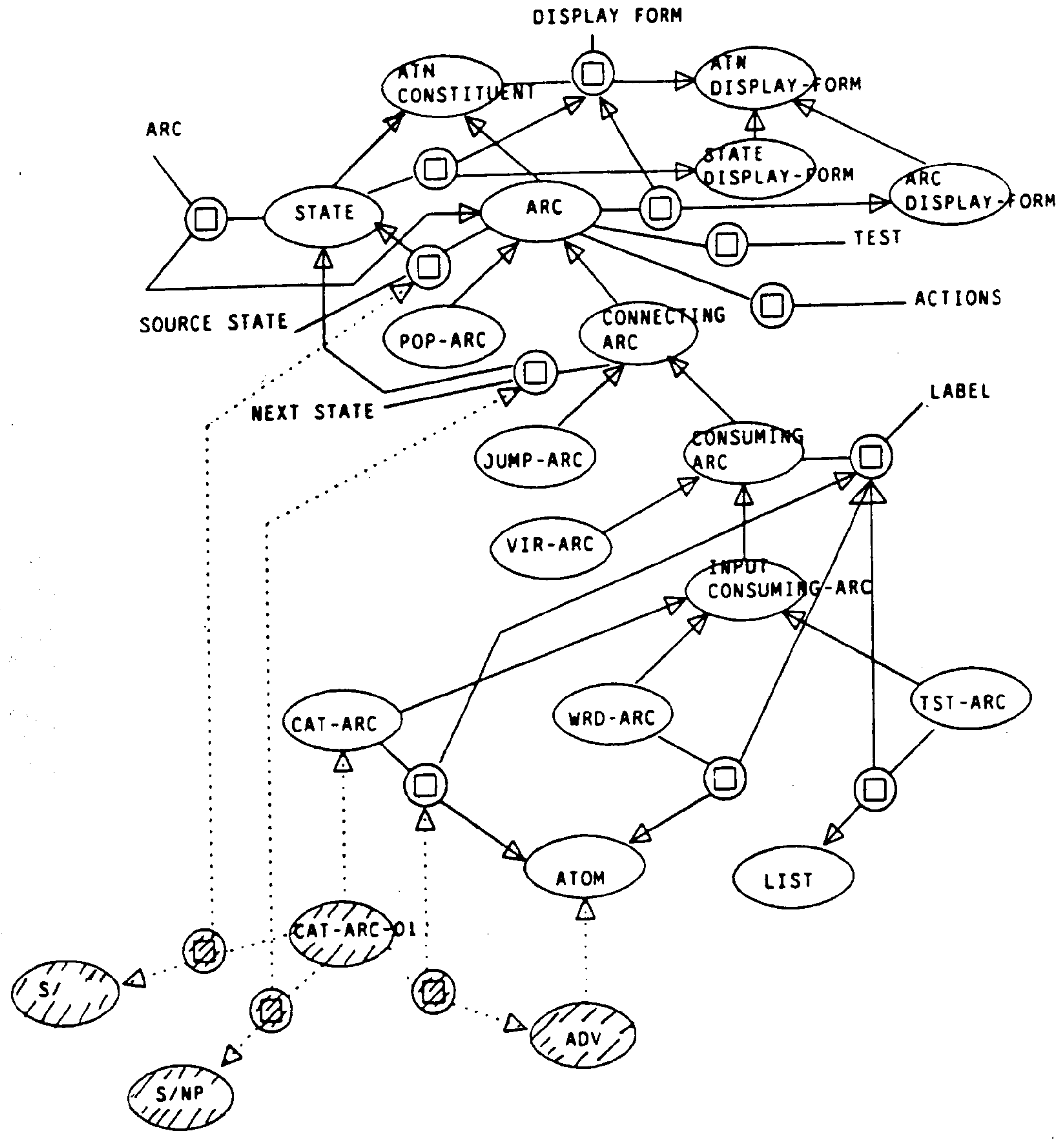

Figure 6. ATN constituents, an example KL-ONE knowledge base.

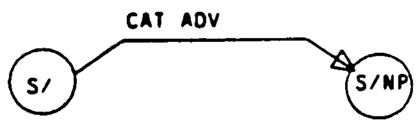

Figure 7. An ATN arc display.

Below the concept [CAT-ARC] is an individual concept representing a particular CAT arc from state S/ to state S/NP. In the figure, individual concepts are represented by shaded ovals, and their corresponding instantiated roles (i-roles) are indicated by shaded role symbols. Dotted arrows connect the i-roles to the roles that they instantiate and to the individual concepts that fill those roles. In the KL-ONE network from which this figure was extracted, this individual concept both represents the object as a component of a grammar (it could be used to answer questions about the grammar or to drive a parser, for example) and also constitutes a displayable object for generating graphical presentations of the object. Using the graphical presentation knowledge from AIPS, an attached procedure inherited by the individual arc concept would draw a picture of the arc whenever a request to draw the individual was generated. The resulting display for the individual arc shown is given in Figure 7.

The Importance of Taxonomic Organization

The ability to organize relevant knowledge in a way that makes it usefully applicable to the problem at hand is a key to many knowledge-based applications. It is one thing to have knowledge in the way that an encyclopedia has it—merely written down somewhere inside, perhaps indexed by subject matter in some way. It is another thing to have assimilated knowledge in such a way that it affects one's perceptions and is fully exploited in problem solving and action. One way to achieve the latter in a computer system is to organize the knowledge of what to do in different situations in such a way that a reasoning engine can automatically recognize when knowledge is applicable. Taxonomic organization is a key technique for addressing such problems. A taxonomic structure can be organized to efficiently locate matching rules in response to a description of a goal. This is one reason why simple taxonomic hierarchies are used in object-oriented programming systems to organize inheritance of methods for code reusability. Similar motivations drive much of the work in KL-ONE and its successors, although with a much greater emphasis on the conceptual structure of the knowledge.

It is not difficult to find matching knowledge for a given task when the entire body of knowledge is small enough to search exhaustively. The challenge comes when the body of knowledge becomes large, and the relevant knowledge must be efficiently extracted from the irrelevant within time and resource limitations. While unable to solve the problem entirely, taxonomic classification techniques as in KL-ONE have important applications to this problem. They are useful not only for efficient organization and retrieval, but also for managing the evolution of large knowledge bases [20] and for managing conflict resolution among simultaneously matching rules [28,37].

For example, when conflicts exist among the rules in a classical rule-based production system, the conflicts may not be discovered until a conflicting situation occurs as input. In a taxonomic classification structure, however, the subsumption of the conditions of one rule by another and the potential for two rules to have common instances can be automatically discovered when the rules are assimilated into the taxonomy. At this time, the person entering the rules can address the question of how two conflicting rules should interact and express the answer in the form of another more specific rule. The advice associated with the more specific rule can then explicitly include a directive to override or alternatively to supplement the more general ones. Such a scheme was implemented in CONSUL [5–8], as an adjunct to KL-ONE, and is incorporated as a part of LOOM [37–39], a successor to KL-ONE. Woods [28] discusses the issue further.

Assimilating production rules into a taxonomic knowledge structure not only facilitates the discovery of interactions at input time, but also promotes a compactness in the specification of the rules. By relying on the fact that concepts inherit information from more general concepts, one can usually create the concept for the pattern part of a new rule by merely adding a restriction to an existing concept. In KL-ONE, when one wants to create a description of a situation that is more specific than a given one, it is only necessary to mention those attributes being modified or added. It is not necessary to copy the attributes of the general situation. Besides facilitating compact memory storage, this feature also facilitates updating and maintaining the consistency of the knowledge base by avoiding the creation of duplicate copies of information.

2.5. Experience with KL-ONE

KL-ONE is primarily a system for organizing conceptual structures into a taxonomy, and as such deals almost exclusively with structural rather than assertional information. The first implementation of KL-ONE used attached data and procedures (much in the spirit of an object-oriented programming language) in conjunction with its nexus and context mechanism to deal with assertional information, but its primary focus was on the taxonomic conceptual structures. This system was successfully used to drive not only portions of a language understanding project [34], but also the AIPS graphical presentation project [31–33] described above. AIPS used KL-ONE as a conceptually structured object-oriented language for expressing knowledge about graphical entities such as shapes, coordinate systems, and coordinate system transformations, and also information about how different kinds of entities should be graphically displayed and represented.

KL-ONE was also used by the CONSUL group at USC/ISI [5–8] to represent knowledge about the structure and function of a computerized message handling system and to provide a mapping between the designers' view of how the system was structured and a user's view of what he or she wanted done. Such knowledge could be used to explain to a user how to achieve his or her goals by using the operators of the system. Key to the CONSUL uses of KL-ONE were not only the organization of the KL-ONE network, but also the action of a "realizer" that would recognize which individuals were described by a concept in a situation, and a "mapper" that would transform one description of an action into another via the rules described earlier. The operation was to repeatedly apply mapping transformations that replaced portions of a stated user goal with corresponding descriptions of system actions until an executable action resulted. At each step in the transformation, the resulting action description was reclassified with respect to the taxonomy to determine what new transformations would become applicable [5].

Experience using KL-ONE in these contexts led to considerable appreciation of the value of taxonomic classification for such problems, but there was also confusion about the nature of individual concepts and an awareness that, although the goals of KL-ONE included a well-defined semantics, a sufficiently formal semantics was not provided, and some of the classifier's operations were not semantically justified. There were many loose ends to be tied down, limitations in expressive power, and subtleties that were not yet understood. Moreover, the original assertional mechanisms were ad hoc and unsatisfying. Attempts to clean up the language and augment its descriptional and assertional power led to the development of a host of successor systems, to which we now turn.

2.6. Successors to KL-ONE

The operation of classification in KL-ONE and its focus on representational issues such as seeking a well-defined semantics and a separation of definition from assertion stimulated a prolific investigation of systems based on the idea of "terminological subsumption." In such systems, a distinction is made between a "terminological component" (or "T-box"), which formalizes the structure of conceptual terms that can be used as constituents of facts and rules, and an "assertional component" (or "A-box"), which records facts and rules composed from those terms. This formal separation of T-box and A-box into separate components was first proposed by Levesque and Brachman in the 1981 KL-ONE Workshop [40]. The terminological component is responsible for certain specialized kinds of reasoning that follow from the structure of the terms, while the assertional component is responsible for general reasoning. Such combinations of specialized inferential components for different kinds of reasoning are referred to as "hybrid" reasoning systems [41].

Several hybrid representational systems based on KL-ONE have been developed and many other systems have explored similar or related themes. Table 1 lists some of these systems, with references. In subsequent sections, we will discuss some of these systems in more detail in the context of various research themes. While we are attempting to give substantial coverage of this work, it is not possible to be complete. There are important pieces of work that we do not have space to discuss. Some of these include KOLA [42,43], micro-KLONE [44], N-ary KANDOR [45], PROTEM and the Intelligent System Server [46–48], QUERELLE [49], QUIRK [50] and QUARK [51], TELOS [52] and VIE-KL [53]. In addition, except for a few illustrations, we have chosen not to address the various applications using KL-ONE and KL-ONE-like systems.

3. A UNIFYING NOTATION

In order to concisely characterize the capabilities of different systems, we will first introduce an expressive terminological language called $\mathcal{KL}$, which is a superset of most of the other languages

Table 1. Some successors to KL-ONE.

- BACK (5.3.1.): [54–57]
- CANDIDE (5.3.4.): [58]
- CLASSIC (5.1.4.): [59,60]
- DRL (5.3.4.): [61–64]
- KANDOR (5.1.2.): [65]
- KL-CONC and KL-MAGMA (5.3.4.): [66–69]
- KloneTalk (5.2.2.): [70,71]
- KL-TWO (5.2.1.): [72,73]
- KNET (5.2.2.): [74–76]
- KREME (5.3.4.): [77]
- K-REP (5.3.4.): [78]
- KRYPTON (5.1.1.): [41,79,80]
- L-LILOG (5.3.4.): [81]
- LOOM (5.2.3.): [16,82,83]
- MANTRA (5.1.3.): [84,85]
- MESON (5.3.2.): [86,87]
- NIKL (5.2.1.): [88–91]
- SB-ONE (5.3.4.): [92,93]
- SPHINX (5.3.3.): [94–96]

For each system listed, we give the section where it is discussed (in parentheses) along with citations.

in the KL-ONE family. We will use this notation to describe various systems by identifying the portions of $\mathcal{KL}$ that they include. We borrow this technique from Patel-Schneider, who introduced the language $\mathcal{U}$ for similar purposes [97]. We present the language using the style of [98] by giving a specification of its syntax and a model-theoretic account of its semantics.

One caveat is that not all terminological languages form a proper subset of $\mathcal{KL}$, so near-misses will be noted. Another caveat is that many other languages use a syntax that differs from ours. While this may lead to some confusion, coercion of all these languages into a common syntax greatly simplifies their comparison. In fact, the $\mathcal{KL}$ language includes a number of redundancies that were introduced solely for comparison purposes.

Section 3.1. gives the syntax and semantics of $\mathcal{KL}$. Section 3.2. describes common T-box operations while Section 3.3. describes common A-box operations.

3.1. *Syntax and Semantics*

A specification in $\mathcal{KL}$ consists of:

- a set of concepts,
- a set of roles,
- a set of disjointness assertions among concepts, and among roles,
- a set of individuality assertions about concepts, and
- a terminology, which maps names to specifications of concepts and roles.

The formal syntax of $\mathcal{KL}$ appears in Backus-Naur form (BNF) in Figure 8.

The operators cdef and cprim allow one to specify and name concepts and to "store" those name/specification associations in the terminology. cprim names a *primitive* concept, whose specification represents conditions that are necessary but not sufficient. cdef names a *defined* concept, whose specification represents conditions that are both necessary and sufficient. rdef and rprim work similarly for roles.

For example, assume that the primitive concept named [MAMMAL], which represents all mammals, has already been specified. We can specify [PERSON], which will represent all persons, as (cprim PERSON MAMMAL). This states that [PERSON] is a primitive concept that is subsumed by [MAMMAL] with the informal interpretation that all persons are mammals. Since [PERSON] is primitive, however, we have no sufficient criteria for determining when some mammal is a person.

```
<terminology>      ::= <spec>*
<spec>             ::= <term-intro> | <concept> | <role> | <individual-spec> | <disjoint-spec>
<term-intro>       ::= (cprim <concept-name> <concept>) |
                       (cdef <concept-name> <concept>) |
                       (rprim <role-name> <role>) |
                       (rdef <role-name> <role>)
<concept>          ::= top |
                       <concept-name> |
                       (and <concept>+) |
                       (or <concept>+) |
                       (not <concept>) |
                       (all <role> <concept>) |
                       (some <role>) |
                       (c-some <role> <concept>) |
                       (atleast <minimum> <role>) |
                       (c-atleast <minimum> <role> <concept>) |
                       (atmost <maximum> <role>) |
                       (c-atmost <maximum> <role> <concept>) |
                       (rvm <role> <role>) |
                       (rvm= <role> <role>) |
                       (sd <concept> (<role> <role>)+)
<role>             ::= top-role |
                       <atomic-role> |
                       <role-name> |
                       (and-role <role>+) |
                       (or-role <role>+) |
                       (not-role <role>) |
                       (restr <role> <concept>) |
                       (domain <concept>) |
                       (range <concept>) |
                       (self) |
                       (inv <role>) |
                       (chain <role> <role>+)
<individual-spec> ::= (individual <concept>)
<disjoint-spec>    ::= (disjoint <concept> <concept>+) |
                       (disjoint-role <role> <role>+)
<minimum>          ::= "a non-negative integer"
<maximum>          ::= "a non-negative integer"
<atomic-concept>s, <concept-name>s, <atomic-role>s and <role-name>s are all formal symbols.
We require that they do not overlap so that they are distinguishable by context.
```

Figure 8. Syntax of $\mathcal{KL}$.

We can now specify the concept [GARDENER]:

(cdef GARDENER (and PERSON (c-some Hobby GARDENING-ACTIVITY))).

That is, [GARDENER] is a defined concept that is subsumed by [PERSON] and has at least one filler of the [Hobby] role that satisfies [GARDENING-ACTIVITY]. The informal interpretation is that a gardener is a person who has at least one hobby that entails gardening. In this case, we not only know some things that are necessarily true of gardeners, we also have a sufficient criterion for determining when a person is a gardener.

As a special case, we allow for *atomic* concepts and roles. An atomic concept carries no information other than its name. It is primitive but has no associated conditions. An example would be a concept [Frob], specified as (cprim Frob top). Being subsumed by top places no restrictions at all on [Frob]. Atomic roles can be specified similarly. (Similar specifications using cdef instead of cprim would merely define alternative names for top.)

There is a simple translation from much of $\mathcal{KL}$ to KL-ONE. The concept top is equivalent to the KL-ONE concept [THING], which represents the entire domain. The operations cprim and cdef allow the specification of primitive and defined concepts, respectively, both of which can be specified in KL-ONE. The operation and allows one to specify terms with multiple conditions, including possibly multiple superconcepts, roles, and structural descriptions, which is also allowed in KL-ONE. The operation all is equivalent to a value restriction; atleast and atmost, taken together, are equivalent to a number restriction; and rvm and rvm= are equivalent to role value maps. sd is equivalent to structural descriptions. The only difference between $\mathcal{KL}$ and KL-ONE regarding concepts is that or and not are new and did not exist in KL-ONE. Also, c-atleast represents new expressive power over KL-ONE, allowing one to define, for example, [PROUD-PARENT] as a person with at least one child who is a doctor.

As mentioned earlier, redundancies exist within $\mathcal{KL}$. The some operator is equivalent to atleast with a minimum of 1. The operator c-atleast is equivalent to atleast combined with restr. For example, the following specifications of [PROUD-PARENT] are equivalent.

```
(cdef PROUD-PARENT (and PERSON (c-atleast 1 Child DOCTOR)))
(cdef PROUD-PARENT (and PERSON (atleast 1 (restr Child DOCTOR))))
```

In a similar way, c-atmost is equivalent to atmost combined with range.

The role top-role is analogous to top and represents all pairs of individuals in the domain. KL-ONE had no explicit role taxonomy, and thus no top-most role. In general, the KL-ONE language for describing roles was weak. All roles were effectively primitive and could, optionally, be a differentiation of a more general parent role. Subsumption for roles was thus equivalent to "differentiation" in KL-ONE, but nothing more could be said about roles. Thus, rprim was characterizable, but rdef, and-role, or-role, not-role, restr and inv represent new expressive power. In addition, the operators self and chain could not be used in KL-ONE as roles per se, but could only be used as parts of role value maps or structural descriptions.

We also note that restr is redundant in $\mathcal{KL}$ since it is equivalent to the conjunction of a subsuming role with a range restriction. For example, the role representing offspring who are doctors could be expressed in either of the following ways.

```
(restr Child DOCTOR)
(and-role Child (range DOCTOR))
```

Finally, rvm= is expressible using rvm as the former represents set equality between role fillers and the latter represents a subset relation. For example, the concept representing persons who are their own grandfathers can be specified in either of the following ways.

```
(cdef OWN-GRAMPA (rvm= (chain Child Child) (self)))
(cdef OWN-GRAMPA (and (rvm (chain Child Child) (self))
                      (rvm (self) (chain Child Child))))
```

The model-theoretic semantics for $\mathcal{KL}$ appears in Figures 9 and 10. A model is a pair $< \mathbf{D}, \xi >$ where $\mathbf{D}$ is a domain of individuals and where ξ is an assignment function that assigns a set from $\mathbf{D}$ to each concept and a set of pairs from $\mathbf{D}$ to each role.

Ambiguities arise with respect to terms whose definitions contain cyclic dependencies. An example would be a concept [A], which we could specify, using a role [R], as:

```
(cdef A (all R A))
```

An appropriate model $< \mathbf{D}, \xi >$ must satisfy:

$$\xi[\mathrm{A}] = \{x \in \mathbf{D} | \forall y (< x, y > \in \xi[\mathrm{R}] \rightarrow y \in \xi[\mathrm{A}])\}$$

Unfortunately, this equation can have multiple solutions. Two solutions are $\xi[\mathrm{A}] = \emptyset$ and $\xi[\mathrm{A}] = \mathbf{D}$. Other solutions are possible depending on $\xi[\mathrm{R}]$. Our stated semantics allows for any of these solutions.

$\mathcal{KL}$ includes almost all of the operators used in the various KL-ONE descendants. In a $\mathcal{KL}$ knowledge base, let C be the set of concepts and R the set of roles in a terminology. A model is a set D and an assignment function ξ such that $\xi : \mathbf{C} \longrightarrow 2^{\mathbf{D}}$, $\xi : \mathbf{R} \longrightarrow 2^{\mathbf{D}^2}$ where $2^{\mathbf{D}}$ is the powerset of the domain D, where $\mathbf{D}^2 = (\mathbf{D} \times \mathbf{D})$ and where ξ must satisfy the conditions below:

1. concepts:
   - $\xi[\mathsf{top}] = \mathbf{D}$
   - $\xi[(\mathsf{and}\ c_1 \ldots c_n)] = \bigcap_{i=1}^{n} \xi[c_i]$
   - $\xi[(\mathsf{or}\ c_1 \ldots c_n)] = \bigcup_{i=1}^{n} \xi[c_i]$
   - $\xi[(\mathsf{not}\ c)] = \mathbf{D} - \xi[c]$
   - $\xi[(\mathsf{all}\ r\ c)] = \{x \in \mathbf{D} | \forall y (< x, y > \in \xi[r] \rightarrow y \in \xi[c])\}$
   - $\xi[(\mathsf{some}\ r)] = \{x \in \mathbf{D} | \exists y < x, y > \in \xi[r]\}$
   - $\xi[(\mathsf{c\text{-}some}\ r\ c)] = \{x \in \mathbf{D} | \exists y (< x, y > \in \xi[r] \wedge y \in \xi[c])\}$
   - $\xi[(\mathsf{atleast}\ n\ r)] = \{x \in \mathbf{D} | \exists n \text{ distinct } y\text{'s} < x, y > \in \xi[r]\}$
   - $\xi[(\mathsf{c\text{-}atleast}\ n\ r\ c)] = \{x \in \mathbf{D} | \exists n \text{ distinct } y\text{'s} (< x, y > \in \xi[r] \wedge y \in \xi[c])\}$
   - $\xi[(\mathsf{atmost}\ n\ r)] = \{x \in \mathbf{D} | \neg(\exists n+1 \text{ distinct } y\text{'s} < x, y > \in \xi[r])\}$
   - $\xi[(\mathsf{c\text{-}atmost}\ n\ r\ c)] = \{x \in \mathbf{D} | \neg(\exists n+1 \text{ distinct } y\text{'s} (< x, y > \in \xi[r] \wedge y \in \xi[c]))\}$
   - $\xi[(\mathsf{rvm}\ r_1\ r_2)] = \{x \in \mathbf{D} | \forall y < x, y > \in \xi[r_1] \rightarrow < x, y > \in \xi[r_2]\}$
   - $\xi[(\mathsf{rvm{=}}\ r_1\ r_2)] = \{x \in \mathbf{D} | \forall y < x, y > \in \xi[r_1] \leftrightarrow < x, y > \in \xi[r_2]\}$
   - $\xi[(\mathsf{sd}\ s\ (r_1^c\ r_1^s) \ldots (r_n^c\ r_n^s))] =$
     $$\{x \in \mathbf{D} | \exists y (y \in \xi[s] \wedge \forall z_1, \ldots, z_n ((< x, z_1 > \in \xi[r_1^c] \leftrightarrow < y, z_1 > \in \xi[r_1^s]) \wedge \ldots \wedge (< x, z_n > \in \xi[r_n^c] \leftrightarrow < y, z_n > \in \xi[r_n^s])))\}$$
2. roles:
   - $\xi[\mathsf{top\text{-}role}] = \mathbf{D}^2$
   - $\xi[(\mathsf{and\text{-}role}\ r_1 \ldots r_n)] = \bigcap_{i=1}^{n} \xi[r_i]$
   - $\xi[(\mathsf{or\text{-}role}\ r_1 \ldots r_n)] = \bigcup_{i=1}^{n} \xi[r_i]$
   - $\xi[(\mathsf{not\text{-}role}\ r)] = \mathbf{D}^2 - \xi[r]$
   - $\xi[(\mathsf{restr}\ r\ c)] = \{< x, y > \in \xi[r] | y \in \xi[c]\}$
   - $\xi[(\mathsf{domain}\ c)] = \{< x, y > \in \mathbf{D}^2 | x \in \xi[c]\}$
   - $\xi[(\mathsf{range}\ c)] = \{< x, y > \in \mathbf{D}^2 | y \in \xi[c]\}$
   - $\xi[(\mathsf{self})] = \{< x, x > \in \mathbf{D}^2\}$
   - $\xi[(\mathsf{inv}\ r)] = \{< x, y > \in \mathbf{D}^2 | < y, x > \in \xi[r]\}$
   - $\xi[(\mathsf{chain}\ r_1 \ldots r_n)] =$
     $$\{< x, y > \in \mathbf{D}^2 | \exists z_1, \ldots, z_{n-1} (< x, z_1 > \in \xi[r_1] \wedge < z_1, z_2 > \in \xi[r_2] \wedge \ldots \wedge < z_{n-1}, y > \in \xi[r_n])\}$$

Figure 9. Model-theoretic semantics of $\mathcal{KL}$—Part I of II.

Nebel [99,100] investigated the impact of such cycles with respect to several methods for specifying a semantics. The one we have used is a case of what he calls *descriptive* semantics, but we make no claims that this is the best treatment. Other possible choices are the least and greatest fixed point methods. One of Nebel's observations is that the fixed point methods are deterministic, whereas the descriptive method is not. By "deterministic" we mean that, given a terminology and legitimate assignments for all primitive terms, then the assignments for defined terms are uniquely determined by the semantics of the operators and the assignments for the primitive terms. This is not true of the descriptive method, as demonstrated by our example concept [A]. As a result, the descriptive semantics tends to be weaker than the fixed point methods (i.e., justifies a subsumption conclusion in fewer cases), since in the descriptive case there are more allowable models that must satisfy the subsumption requirement, making subsumption more difficult to conclude. Nebel shows that the greatest fixed point method has the advantage of leading to a conceptually simple subsumption algorithm, but this only applies to simple terminological languages. He also found that each method produces counter-intuitive results on certain examples, either by concluding a subsumption that it should not, or by failing to conclude one that it should. Nebel concludes that the choice of the best method is not obvious. Furthermore, he shows that the unrestricted use of cycles can lead to undecidability.

3. term introductions:
   - $\xi[n_C] \subseteq \xi[c]$ whenever (cprim n_C c)
   - $\xi[n_C] = \xi[c]$ whenever (cdef n_C c)
   - $\xi[n_R] \subseteq \xi[r]$ whenever (rprim n_R r)
   - $\xi[n_R] = \xi[r]$ whenever (rdef n_R r)
4. individuality assertions:
   - $|\xi[c]| \leq 1$ whenever (individual c)
5. disjointness assertions:
   - $\xi[c] \cap \xi[c'] = \emptyset$ whenever (disjoint $c_1 \ldots c_n$) and there exists i and j with $1 \leq i,j \leq n$, $i \neq j$, $c = c_i$ and $c' = c_j$.
   - $\xi[r] \cap \xi[r'] = \emptyset$ whenever (disjoint-role $r_1 \ldots r_n$) and there exists i and j with $1 \leq i,j \leq n$, $i \neq j$, $r = r_i$ and $r' = r_j$.

Note that $|s|$ denotes the cardinality of the set s.

Figure 10. Model-theoretic semantics of $\mathcal{KL}$—Part II of II.

3.2. *Terminological Operations*

A knowledge representation system in the KL-ONE tradition supports a variety of terminological and assertional operations. In this section and the next we will introduce and give definitions for the most common terminological and assertional operations, using the model-theoretic account of the $\mathcal{KL}$ language, where practical. We will also give a general idea of how systems differ with respect to these operations. We will identify the operations offered by individual systems when we discuss them in later sections.

The primary terminological operations are those for specifying terms: the operators cprim, cdef, rprim and rdef. All the systems we will discuss include operations equivalent to the cdef operator. The inclusion of cprim, rprim and rdef varies from system to system. All the systems allow the use of atomic concepts and roles.[9]

Other common terminological operations are:

SUBSUMPTION: In the KL-ONE tradition, a term is said to *subsume* another term if and only if the set it denotes includes the set denoted by the other in every allowed model. In other words, A subsumes B if and only if for every model $< \mathbf{D}, \xi >$, $\xi[B] \subseteq \xi[A]$.[10] *Classification* is the process of identifying all appropriate subsumption relations among a given set of terms.

INHERITANCE: *Inheritance* is the process of identifying those conditions that apply to a concept because they are conditions for a subsumer of the concept.

COMPLETION: *Completion* is the process of identifying and recording all conditions that apply to a concept. This includes all the conditions found through inheritance plus certain other conditions that are logically implied. For example, if a concept representing a hand uses an rvm= to state that the [Finger]s of a [HAND] are the same set as its [Digit]s, then any value restriction for [Finger] of a [HAND] also applies to [Digit] of a [HAND], and vice versa. This information would not be noticed by inheritance per se. Completion would note it and record it explicitly.

[9]Those systems without cprim or rprim allow specifications of atomic terms in other ways.

[10]But see [28] for an argument that the "if" part of this definition is too strong.

COHERENCE: A term is *coherent* if and only if there is a model in which the term's denotation is not empty. In other words, A is coherent if and only if there exists a model $< \mathbf{D}, \xi >$ such that $\xi[A] \neq \emptyset$.

All systems examined in this paper compute inheritance and subsumption. Most perform classification, many check terms for coherence, and some perform completion. In addition, those that allow disjointness specifications usually can determine whether any pair of terms are disjoint.

As we will discuss in Section 4., a major research theme in the KL-ONE tradition has to do with a tradeoff between the expressive power of the knowledge representation system and the soundness, completeness, and computational tractability of its algorithms. Soundness and completeness refer to the fidelity of the algorithms to a model-theoretic criterion like the model-theoretic semantics presented here for $\mathcal{KL}$. Informally, an algorithm is *sound* if it is guaranteed to conclude something only if that conclusion is justified by the model-theoretic semantics—usually, if it is true in all allowable models. Conversely, an algorithm is *complete* if it is guaranteed to draw any conclusion that is so justified. This can best be illustrated by considering the operation of subsumption.

Let SUBS?(A, B) represent a subsumption algorithm that is total (i.e., always returns an answer) and returns either true or false when A and B are terms. SUBS? is *sound* if and only if whenever SUBS?(A, B) returns true, then A subsumes B according to the model-theoretic criterion. SUBS? is *complete* if and only if whenever A subsumes B, then SUBS?(A, B) returns true. If SUBS? is only sound but not complete, then returning true implies that A subsumes B but returning false means that the algorithm cannot tell. If SUBS? is both sound and complete, then returning true implies that A subsumes B and returning false implies A definitely does not subsume B (i.e., there exists a model where $\xi[B] \not\subseteq \xi[A]$).

In nearly all the systems studied, the algorithms that implement the terminological operations are sound, total and computationally tractable. Inheritance is fairly simple, so the inheritance algorithms are usually complete as well. However, the other operations may or may not be complete depending on the system.

3.3. Assertional Operations

The assertional operations provided by KL-ONE-like systems vary from system to system much more than do the terminological ones. We will give only a brief review of their differences. All A-boxes we will examine provide basic database storage and retrieval operations. Namely, one can introduce individual constants and record the concepts they realize and values for their roles. In addition, one can query the A-box to identify all individual constants that realize a concept or role, or that relate to a given individual constant via a given role. Some systems allow one to "*close*" a role for a given individual, i.e., to assert that all of its fillers are known, so that one can conclude that anything not known to be a filler is not one. Moreover, some offer more sophisticated assertional capabilities—e.g., SPHINX allows Prolog-like rules to be asserted—and others offer sophisticated retrieval languages—e.g., LOOM's retrieval language incorporates a full first-order language. Some determine consistency, and some provide truth-maintenance facilities with retraction. Finally, many perform the following operation of *realization*, which identifies individuals that can be described by a concept in a situation.

Realization is the process of identifying all concepts that are realized by an individual—i.e., all the concepts of which it is an instance. The operation of realization applies to individual constants, which are usually found in the A-box, and are not to be confused with individual concepts (which, if they are permitted, are found in the T-box and in this notation, as in NIKL, are like definite descriptions denoting sets that are either singleton or empty.) A concept *is realized by* an individual constant if and only if, in every model, the interpretation of the individual constant in the model is in the set denoted by the concept. Since we have not introduced individual constants into our model theory (indeed, we have not addressed the model-theory of the assertional component at all), we will not offer a formal definition of "realize."

The process of realization is affected by whether or not the *closed world assumption* is being made. For example, if [CHILDLESS-PERSON] is a concept denoting people without children defined by

(cdef CHILDLESS-PERSON (and PERSON (atmost 0 Child)))

and if we have an individual constant, *p*, about which we only know that it represents a person, can we conclude that *p* represents a [CHILDLESS-PERSON]? The answer is "yes" if we make the closed world assumption and "no" if not.

None of the systems examined in this paper that perform realization make the closed world assumption as a general rule. However, as mentioned above, some have special facilities to "close" a role. For example, in CLASSIC, the above *p* would not be realized as a [CHILDLESS-PERSON] unless one *closed* the [Child] role for *p*.

4. MAJOR RESEARCH THEMES

The strengths of KL-ONE led several researchers to develop similar knowledge representation (KR) systems with some variations. Much effort was expended in making refinements to KL-ONE and in exploring developments of and variations on its themes. In the remainder of this paper, we present some of the themes of this research.

An important landmark was the development of KRYPTON, which identified three key issues for KL-ONE-related research: the separation of definitional from assertional information, the specification of a formal semantics, and the trade-off between expressive power and computational tractability. In this section, we will consider each of these issues, along with several other important themes.

4.1. Distinguishing Definitions from Assertions

A major issue raised by KRYPTON was that definitions and assertions should be clearly separated. This was first raised for semantic networks by Woods [1], as mentioned earlier, and introduced in the KL-ONE context by Brachman and Levesque [15,101]. KRYPTON took an extreme position by separating the two kinds of information into distinct components. Definitions went into a T-box while assertions went into an A-box. The two boxes were implemented in KRYPTON as separate modules with separate data structures. Information flowed between them in a limited and carefully specified manner (we examine this further in Section 4.3).

The conceptual separation of definitions and assertions has played a part in every KL-ONE-like system, including KL-ONE itself. While the designers of KRYPTON took an absolute position on the matter, designers of other systems were willing to blur this distinction. One problem is that there does not appear to be a commonly accepted meaning for "assertion," and often its use confuses as much as it clarifies [16]. In particular, several types of information seem to cross the definition/assertion boundary.

One such type of information includes terms for which no complete accounting can be made, but for which certain necessary conditions apply. An example is a term for the class of all persons. All persons are necessarily mammals, among other things, yet it is commonly accepted that the class of persons forms a natural kind, and no complete definition of "person" can be constructed. One could specify a concept to represent persons in KL-ONE using cprim, as we did earlier, to specify [PERSON] as a primitive concept subsumed by the primitive concept [MAMMAL]. While this is in some sense "definitional" in that the symbol [PERSON] has now been introduced to refer to the class of persons, it is not a complete definition since it does not provide both necessary and sufficient conditions. In particular, it does not specify what differentiates persons from mammals. But it seems also to be partly assertional in that it represents the fact that all persons are mammals. Although it did not do so originally (see PrimGeneric in [79]), KRYPTON eventually rejected this blur by not allowing primitive terms to have any structure, i.e., a primitive term could not have necessary conditions associated with it. [PERSON] and [MAMMAL] could be primitive concepts in KRYPTON, but the fact that all persons are mammals would normally be represented in the A-box.[11]

[11]With some effort, one could represent [PERSON] in a different way that would capture its subsumption by

The more common view, which most systems in the KL-ONE family adopt, and which we incorporated into $\mathcal{KL}$, is simply to take the position that constructs that introduce or specify terms should be distinguished from other facts, whether they provide complete definitions or not. Some terms are specified as definitions (with conditions that are both necessary and sufficient), while others are specified as primitives (with conditions that are necessary only).

Another problem with the definition/assertion distinction is that often it does not go far enough. One reason for distinguishing term specifications from assertions is the assumption that the former are always necessary (if not always sufficient). If contradictions arise, such necessary information is not subject to being wrong and recovery procedures can thus be directed elsewhere. However, there is a subtle confusion here between necessary in the sense of being part of the meaning of the term, versus necessary in the sense of being necessarily true in all cases. The former notion, that of conditions which are part of the meaning of a term and therefore true "by definition," is sometimes called "analytic" in philosophical discussions. It turns out that there can be conditions that are necessarily true but are not part of the meaning of a term (i.e., are not analytic). One class of such conditions includes the necessary consequences of a term's definition. For example, it is necessarily true that a polygon with three sides will have three angles, but this is a consequence of, rather than part of, the definition of a triangle. As Woods [28] points out, if one were using a KL-ONE-like system and wanted to capture in the terminology the facts that triangles are the same class as polygons with three sides and polygons with three angles, then both would need to be part of the definition of a triangle. Unfortunately, if [TRIANGLE] is defined as:

```
(cdef TRIANGLE (and POLYGON (atleast 3 Side) (atmost 3 Side)
                            (atleast 3 Angle) (atmost 3 Angle)))
```

then the information is lost that either polygon with three sides or polygon with three angles would be sufficient to conclude that something is a triangle and certain subsumption relationships are likely to be missed. Clearly, one would like to say that a polygon either with three sides or with three angles is a triangle, and that all triangles necessarily have both three sides and three angles. Only a few KL-ONE-like systems examined herein can state all of this, and they would do so for triangles by using the above cdef specification plus a statement of the appropriate sufficiency conditions, to which we now turn.[12]

A similar issue arises for information that is sufficient but not necessary. Given a class, there may be several different sets of associated sufficient conditions. For example, an animal whose parents are persons must be a person, and a featherless biped also must be a person. Only a few of the KL-ONE-like systems allow one to express pure sufficiency conditions. For example, LOOM, MESON and CLASSIC have done so using rules.

4.2. *Disjointness*

A second type of information that often crosses the definitional/assertional boundary is disjointness. An example is the fact that the class of persons and the class of dogs are disjoint. An easy method of representation was adopted in NIKL, where disjointness declarations among previously specified concepts could be made in the T-box by simply declaring concepts to be disjoint. However, even though these declarations are made in the T-box, the treatment in NIKL is effectively assertional since, in this case, the specification of neither [PERSON] nor [DOG]

[MAMMAL] in the T-box, but such representations are not the ones that would spontaneously come to mind. For example, one could represent

(cdef PERSON (and MAMMAL PERSONNESS))

where [PERSONNESS] is a (probably primitive) concept representing all individuals having the properties common to persons other than those implied by being mammals.

[12]Woods [28] raises and deals with this issue by distinguishing the operation of definition from other conditional relationships, allowing a concept's necessary conditions to be different from its sufficient conditions, and by allowing some structures to have both definitional and assertional import. The result is a generalized notion of definition that provides for abstract concepts with multiple alternative definitions and for partially defined concepts whose necessary conditions are not the same as their sufficient conditions.

states anything about their being disjoint. That they are disjoint is stated after the terms are introduced.[13] An example of a definitional approach would first specify, say, [PERSON], and then specify [DOG] such that [DOG] is a primitive concept that is subsumed by, among other things, the ***negation*** of [PERSON]. Here, the negation of a concept is interpreted as in $\mathcal{ALC}$ [103], namely it represents all objects that are not in the class denoted by the concept. A minor problem with this definitional approach for representing disjointness is its asymmetry—the specification of [DOG] mentions [PERSON] but not vice versa. While not a serious problem, this led researchers to look for other representational means. For example, in CLASSIC, disjointness is both definitional and symmetrical. It differs from the $\mathcal{ALC}$ scheme above by using user-supplied indexes for collections of disjoint concepts to encode the fact that every pair of concepts in the collection is pairwise disjoint. A minor difficulty is that these disjunction class indices are uninterpreted labels used to encode the relationships among the concepts in the collection, and otherwise have no semantic import.

4.3. *Hybrid Systems*

The separation of definitions and assertions into distinct components in KRYPTON was motivated by more than just a desire to maintain the definition/assertion distinction. It was also considered more efficient to build specialized modules for the sub-languages of a system so that specialized procedures could run over specialized data structures, resulting in efficient components. This approach to KR became quite popular and has been called the ***hybrid system*** approach. Nearly all KL-ONE-like systems after KRYPTON adopted this approach. Moreover, it is not limited to just T and A boxes, but can be used for any number of components (e.g., MANTRA [84,85] has four—see Section 5.1.3.). Other lines of research on hybrid KR systems, such as the CAKE system[104–106], have also been investigated.

In nearly every KL-ONE-like hybrid system, information flows readily between components. In general, if there is information in one component that, if shared with another, would lead to an inference, then specialized routines are implemented to make that information flow between components and to draw the appropriate inferences. For example, if the T-box were told that [MAMMAL] subsumes [PERSON] and the A-box were told that p is a [PERSON], then, if queried, the A-box would conclude that p is a [MAMMAL].

Unfortunately, there are some costs associated with the coordination and flow of information between components of a hybrid system. Some of these specialized inter-component routines are complex and difficult to design. In Section 5.2.1., we examine the difficulties and subtleties encountered in the design of KL-TWO. Such complexities involve trade-offs, and the advantages of a hybrid approach must be weighed against the disadvantages.

Note that it is not necessary to separate definitions and assertions into different components in order to make a definitional/assertional distinction. In CLASSIC, for example, we find that the T and A box distinction is not crucial, even though the distinction between definitions and assertions is maintained (see Section 5.1.4.).[14]

4.4. *Formal Semantics*

An important issue raised by the KRYPTON effort was that a KL-ONE-like language can be viewed as a formal language for which a formal semantic account can be given. In fact, the authors of KRYPTON went further and claimed that such an account ***should*** be given. While this observation was already ingrained in other lines of KR research, such as Hayes' logic of frames [107], Shapiro's SNePS system [108], and Schubert's semantic nets [109,110], Brachman, Levesque and Fikes were the first to do so for the KL-ONE family.

[13]Unfortunately, this treatment of disjointness in NIKL infected the classifier. Since the classifier was ostensibly intended to examine only "definitional" information, it ignored disjointness declarations. Thus, two defined concepts whose only difference was that they were disjoint would be considered equivalent and thus merged. This was the subject of a complaint in [102].

[14]Woods [28] discusses some of the disadvantages of separating terminological and assertional knowledge into separate components and points out some advantages of integrating these two kinds of information more effectively.

At first, Brachman, Levesque and Fikes provided a translation from the language of KRYPTON to first-order predicate calculus, which in turn could be provided with the usual model-theoretic account. Eventually, they chose to assign a model-theoretic semantics directly to the KRYPTON language. This style of account was used in Section 3.

The advantage of providing a semantic account is that it gives an independent standard of correctness—a criterial semantics as discussed earlier. Algorithms that store, retrieve or infer information can be given formal standards to meet and implementations can be compared against those standards. The clarity this provides has been so useful that semantic accounts have become a standard component of every KL-ONE-like system developed after KRYPTON.

4.5. *Limited Expressive Power and Tractability*

One issue raised by KRYPTON that has had the strongest impact on KL-ONE-related research is the tradeoff between expressive power and computational tractability. Following Levesque [111,112], Brachman, Levesque and Fikes made initial arguments in [79], and Brachman and Levesque later made stronger arguments [98,113] that the answers given by a KR system should be fully correct with respect to a formal semantics and that the operations should be performed in a predictable and reasonable amount of time.

These requirements can have very severe consequences for the system's inferential algorithms. A common interpretation was that these algorithms should be sound, complete and tractable. A consequence of this interpretation was that the expressive power of each component of a KR system should be limited, since otherwise a system cannot simultaneously be sound, complete, and tractable for classification and other inferential algorithms. Brachman and Levesque expressed the hope that, by combining various components in creative ways, an expressive and powerful KR system could result that would be made up of sound, complete and tractable sub-parts. This became a key goal of the *hybrid system* approach.

KRYPTON attempted to meet these requirements for the T-box with respect to two types of inference, namely, inheritance and subsumption determination. Inheritance algorithms had long before been developed that met these criteria. However, the complexity of determining subsumption was somewhat of a mystery at the time. (The unraveling of this mystery is summarized in Section 5.4.) To meet these requirements, the expressive power of KRYPTON's T-box language was specifically chosen such that determining subsumption would be tractable.[15] As a result, the T-box language was quite expressively weak as compared to KL-ONE. That is, certain things that could be expressed in KL-ONE could not be expressed in KRYPTON. To balance this weakness, the language for the A-box of KRYPTON was powerful, namely, the first-order predicate calculus. The inferential algorithm offered was that of full deduction, which was based upon Stickel's connection graph theorem prover [114]. Thus, the A-box's main inferential algorithm was sound and first-order complete, but intractable. Here, the goal of tractability was sacrificed for completeness.

The attractiveness of the above functional requirements was clear: if they could be met, predictable answers could be guaranteed in a predictable time. It thus launched a long series of investigations into limited systems that tried to meet these goals.

4.6. *Expanded Expressive Power and Inferential Services*

While some researchers followed the KRYPTON strategy of developing KR systems with limited expressive power and limited inferential services, others chose instead to explore systems with wide expressive power and expanded inferential services. Where the former were motivated by the goals of computational tractability and predictable consequences presented above, the latter adopted a different view—one of service to the user. This approach starts from the assumption that each user of a KR system has certain representational and inferential needs. If the user's needs are not met by the KR system, then the user must go "outside" the system. This can lead to *ad hoc* connections between the KR system and data structures that the user must supply for representing what the KR system will not. In addition, it can lead to *ad hoc* procedures for

[15] Actually, the first design of KRYPTON did not have this goal [79], but successive designs did [80].

drawing inferences since the information is now spread among the KR system's and the user's data structures. Thus, these researchers decided it was better to have greater expressive and inferential power, even though it would preclude having both completeness and tractability. As it turns out, complete and tractable inferential algorithms are nearly impossible to attain for any but the weakest of languages (see Section 5.4.).[16]

In a sense, KRYPTON fits into both the limited and the expansive categories in that KRYPTON's T-box was limited but its A-box had the widest expressive and inferential power of all the A-boxes that we will examine. Systems such as SPHINX offer T-box languages that are slightly weaker than KL-ONE's, but whose A-box languages are relatively strong. In the case of SPHINX, the A-box language was that of Horn clauses. On the far end of the spectrum, LOOM offers the most expressive T-box language of all those examined, along with a host of programming metaphors including rule and objected-oriented programming. Overall, each system has taken a slightly different stand on these issues.

5. THEME DEVELOPMENT

The issues discussed in the previous sections constitute some of the primary themes that have motivated much of the KL-ONE-related research. In the next few sections, we highlight the specific results of various systems that have pursued these themes and a variety of subthemes. We divide our review as follows. Section 5.1. reviews systems that accepted the KRYPTON challenges directly, including KRYPTON itself. Section 5.2. reviews those systems that are more or less direct descendants of KL-ONE. While influenced by KRYPTON, these systems did not take KRYPTON's goals as primary. Section 5.3. examines several other notable KL-ONE-like systems. Section 5.4. reviews the results of analyzing the complexity of subsumption for a host of terminological languages. Finally, Section 5.5. examines some criticisms and dissenting opinions.

5.1. The KRYPTON Challenge

With the challenge of hybrid systems out in the open, a number of researchers chose to design hybrid systems that were expressive, powerful and manageable (though often not tractable, as we will see). To demonstrate this thread of KL-ONE-related research, we examine the following systems: KRYPTON, KANDOR, MANTRA and CLASSIC.

5.1.1. KRYPTON

In designing KRYPTON [41,79,80], Brachman, Levesque and Fikes wanted the overall KR system to have expressive power that went well beyond that of KL-ONE (at least on the assertional side). Of particular concern was the representation of incomplete knowledge such as "one of the three people in that room is a murderer" or "there is a cow in the field." The first-order predicate calculus could handle these with

$$Inroom(A,R) \wedge Inroom(B,R) \wedge Inroom(C,R) \wedge (Murderer(A) \vee Murderer(B) \vee Murderer(C))^{17}$$

and

$$\exists c(Cow(c) \wedge Infield(c,F)).$$

Thus, they chose a first-order language as their representational formalism. However, they also wanted to take strong advantage of KL-ONE-like taxonomies. Strengthening the definitional/assertional distinction that was partially addressed in KL-ONE, and applying their strong desiderata regarding the functional role of a KR system, they adopted the hybrid T-box/A-box design that we have already discussed in Section 4.

[16]For some systems, certain inferences, such as determining subsumption, is merely co-NP-hard (e.g., BACK), while for others, it is undecidable (e.g., NIKL).

[17]But note that this does not give an account of an individual concept that would be described as "the murderer," but whose identity remains unknown. This was one of the goals which KL-ONE's nexus mechanism attempted to solve and is still an open problem.

KRYPTON's T-box was strictly definitional with weak expressive power, and its A-box was strictly assertional with strong expressive power. Initially, the T-box language was designed to include the concept operators and, all, atleast, atmost, disjoint, cprim and cdef, and the role operators restr, chain, disjoint-role, rprim and rdef [79] (see our Figures 8, 9 and 10). However, the T-box language that was actually implemented was considerably smaller due to the desire to keep subsumption tractable, coupled with time limitations on the project. In the end, the T-box language consisted of the concept operators and, all and cdef, and the role operators chain and rdef [80].

The A-box offered a full first-order language and used deduction to answer queries. Thus, queries to the A-box took an unpredictable amount of resources given the semi-decidability of deduction in a full first-order system. Deduction in the A-box was performed by a refutation-based connection-graph theorem prover developed by Stickel [114]. Information flow between the T and A boxes went primarily from T to A and was expressed primarily by extending the power of unification based on Stickel's partial theory resolution. For example, if [PERSON] was subsumed by [MAMMAL], then unification was extended by allowing (PERSON x) to unify with (MAMMAL y). Since the theorem prover was based on a connection graph, these extended unifications were identified by having the T-box add them to the connection graph. See [80] for further details.

The KRYPTON system was a landmark for the issues it highlighted and on which it took a firm stand. However, due to its particular combination of a weak terminological component and an unwieldy assertional component, it did not find its way into applications. To address this, midway into its development, the KRYPTON group began the development of KANDOR.

5.1.2. KANDOR

The goals of KANDOR [65] were the same as those stated earlier for KRYPTON. KANDOR's T-box language was chosen so as to be an expressive subset of KL-ONE with the property that subsumption was tractable. KANDOR's T-box language included the concept operators and, all, atleast, c-atleast, atmost, disjoint, cprim and cdef plus the role operators range and rprim. KANDOR's A-box was effectively a database, offering the basic A-box operations. The inferential algorithm offered by the A-box was realization. It did not make the closed world assumption. Realization appears to be tractable in KANDOR.[18] Unfortunately, it was later shown that complete subsumption in KANDOR is co-NP-hard [115] and, consequently, that KANDOR's classifier is incomplete. KANDOR did find its way into applications, most notably the ARGON system [116]. ARGON eases query processing into a database for casual users by allowing incremental construction of queries, guided by a taxonomy.

5.1.3. MANTRA

Beginning in the late 1980's, Bittencourt began the development of the MANTRA system [84,85]. MANTRA combines four components: a first-order logic, a KL-ONE-like term component, a semantic net component with defeasible inheritance, and a production rule system. All features of MANTRA were chosen with two goals in mind: each should be semantically motivated and all algorithms involved should be decidable. The first three components form the basis of the knowledge representation facility upon which the production system operates.

To meet the decidability requirement, the first-order language (the *logic* box) uses a four-valued semantics, based primarily upon [117], that was specifically chosen so that entailment involved no chaining (see Section 5.4.). The four values are the powerset of $\{true, false\}$, i.e., $\{true\}$, $\{false\}$, $\{true, false\}$, and $\emptyset$. The first two values correspond to true and false, respectively. The third value corresponds to having contradictory evidence both for and against a given proposition, while the value $\emptyset$ corresponds to having no information about a given proposition. The logic box performs complete deduction with respect to its semantics and is decidable. A consequence, however, is that this non-standard semantics reduces its utility for some users (see the comments at the end of Section 5.4.). The term language (called the *"frame*

[18]No proof is offered but the tractability of realization is argued for in [65].

box") includes and, or, all, exists, cprim and cdef for concepts. Instead of roles, MANTRA's frame box introduces *relations*, which are n-ary relations with $n > 1$. It offers n-ary versions of the and-role, or-role, domain, range, rprim and rdef operators. Subsumption is the only inferential algorithm performed by the frame box, and is decidable. The semantic net language (the *snet* box) is based upon [118] and allows nodes to be connected by default or exception links and answers questions about subsumption between nodes. Of course, the names of terms can overlap in the three boxes and thus information about terms can appear throughout the system.

Bittencourt gives compatible four-valued semantics to each of these three sub-languages. Furthermore, he defines the questions that each can be asked along with what constitutes the correct answers. He then defines the questions that can be asked from pairs of these sub-languages along with what constitutes the correct answers. He thus has a clear, formal characterization of the entire hybrid system, along with a working implementation.

5.1.4. CLASSIC

The most recent work in this line of development is the CLASSIC system [59,60], now being developed at AT&T Bell Laboratories. The developers of CLASSIC include Brachman and Patel-Schneider, two of KRYPTON's developers. Here, we find that certain traits of KRYPTON have remained: CLASSIC was designed to have tractable subsumption but with greater expressibility. We also find that some traits are gone: there is no physical separation of the T and A boxes. In CLASSIC, the language for describing terms is the same as the language for describing individuals. The definition versus assertion distinction remains, but it is less clear than in KRYPTON given the lack of separate T and A boxes. The classifier only pays attention to definitional operators. The cprim operator, which some would argue carries assertional import, is considered by the classifier. However, Brachman *et al* argue that cprim is entirely definitional. On the other side, CLASSIC allows rules of the form "if an individual is of type A then it is also of type B", which amount to sufficiency conditions (rules of this form first appeared in LOOM, Section 5.2.3., and second in MESON, Section 5.3.2.). These rules can provide sufficient conditions for a concept. However they are not considered by the classifier and are thus considered assertional. Overall, the goals of CLASSIC appear to be more pragmatic than those of KRYPTON. CLASSIC's designers have taken advantage of their experience regarding user needs, implementational methods, complexity measures, and the importance of certain distinctions. The result is a tractable and reasonably expressive KR system.

The T-box language includes the operators and, all, atleast, atmost, disjoint,[19] cprim and cdef for concepts and no operators for roles. In addition, CLASSIC has an rvm= operator where the roles can use chain. However, all roles and role chains appearing in an rvm= must be single-valued. Like BACK, CLASSIC allows a concept to be defined by an explicit, extensionally defined set (using an operator called one-of). CLASSIC also allows a concept to be defined by an individual filling a given role using an operator called fills. It also includes a concept-forming operator, test, that takes a function as its argument to be applied as a predicate against individuals. To the T-box, test concepts are primitive. However, the test function can be applied to individuals to see if they instantiate the concept. This is useful for concepts whose criterion is best described procedurally. However, it has the disadvantage that no analysis of these test functions is possible for use by the classifier. Finally, as mentioned above, one can express rules of the form $A \Rightarrow B$ where A and B are concepts.

At the time of this writing, the authors of CLASSIC argue informally that subsumption in CLASSIC is indeed tractable, however no proof is yet available. Interestingly, the authors are considering expansions of the term language to include role operators such as and-role, inv, and rdef. These are needed by certain applications and may be added even though this would most likely preclude tractable subsumption.[20]

The system performs classification, completion, coherence checking and realization. In addition, it executes rules against individuals (as appropriate) in a forward chaining manner. It also

[19]Disjoint declarations in CLASSIC are actually specified in a definitional fashion, not in the assertional fashion of $\mathcal{KL}$.

[20]Personal communication with R. J. Brachman, May 1990.

detects contradictions and allows retractions. Whenever an assertion is made, the system uses forward reasoning to conclude all it can. If a contradiction is detected, the most recent assertion is retracted and the user is informed.

5.2. Direct Descendants of KL-ONE

Beginning with NIKL [88–91], several systems followed in the footsteps of KL-ONE. While respecting and partially adopting the design goals of KRYPTON, these systems were more concerned with providing greater expressivity and less concerned with completeness and tractability issues. The systems we will review are NIKL, KL-TWO, KNET, KloneTalk and LOOM.

5.2.1. NIKL and KL-TWO

In 1982, the developers of KL-ONE who remained at Bolt Beranek and Newman and at USC/Information Sciences Institute undertook a new implementation of KL-ONE that eventually became NIKL [88–91]. The goals in developing NIKL were (1) to improve the utility of the language while retaining many of its useful features and (2) to make it more efficient. NIKL also provided a base for research in hybrid systems, leading to the KL-TWO system [72,73] that we discuss below.

The design of NIKL was strongly influenced by the work that led to KRYPTON. The assertional component of KL-ONE, consisting of nexuses, description wires and contexts, was discarded on the grounds that (1) it was both awkward and expressively weak, and (2) NIKL would address only the definitional side of the definition/assertion separation (although, as we will see, certain assertional constructs remained in NIKL). A separate role hierarchy was introduced, which was the culmination of role differentiation in KL-ONE. Finally, a formal semantics was provided. For assertions, a new and separate component would be designed, which eventually became the component known as PENNI (again, see below).

An interesting historical note is that, initially, structural descriptions were discarded because it was thought that their expressive power was redundant given the rvm= operator plus the (newly introduced) notion of defined roles. Later on, around the time that a formal semantics was adopted, it became clear that structural descriptions could not be captured using other constructs and, eventually, they were added to NIKL. They retained the same expressive power as in KL-ONE, which includes a fixed, and thus limited, type of quantification. Most of the KR systems examined in this paper do not include structural descriptions.

In the end, NIKL's language offered and, all, atleast, atmost, rvm, sd, individual, disjoint, cprim and cdef for concepts and and-role, domain, range, inv, rprim and rdef for roles.[21] In addition, when specifying roles within an rvm or sd, one could also use self and chain. NIKL performed classification and completion, and it determined coherence and disjointness. All of these operations were implemented with efficient algorithms, but none were complete.

Along with NIKL's development came the development of what became known as KL-TWO [72,73]. In choosing an assertional language, the developers decided that at least the power of a quantifier-free propositional calculus was necessary. The choice was then made to use the RUP system [119,120] as an A-box. RUP was a truth maintenance system (TMS) that offered a propositional calculus with equality based on a three-valued logic (true, false and unknown) and offered a suite of tools for dealing with contradictions and performing retraction, forward chaining, and demon-like procedure execution. RUP was expanded to a system called PENNI that caused information to flow between NIKL and itself. The entire system was called KL-TWO.

PENNI's responsibilities were primarily (1) to reflect terminological information in the assertional component, and (2) to perform realization. For an example, consider the concept [GARDENER] introduced earlier:

(cdef GARDENER (and PERSON (c-some Hobby GARDENING-ACTIVITY)))

[21]disjoint, individual and inv were ignored by the classifier and were thus more like simple markers rather than term specification operators (although disjoint was used by the disjointness algorithm).

Regarding (1), if PENNI is told that (GARDENER *g*), it would then respond with "true" when asked the truth status of (PERSON *g*). Regarding (2), if instead PENNI is told that (PERSON *p*), (Hobby *p a*) and (GARDENING-ACTIVITY *a*) are all true, it would then respond with "true" when asked the truth status of (GARDENER *p*).

While the initial motivation for KL-TWO arose from a need to expand NIKL, the designers soon realized that the A-box would receive primary use in many applications, rendering the T-box as secondary. The hard part in designing KL-TWO was getting the TMS to work correctly with the classifier. The TMS is designed to draw incremental inferences and to record each one. The classifier, on the other hand, is designed to draw larger scale inferences and not to record them. Bridging these differences so that the TMS worked correctly took considerable effort.

PENNI always made the open world assumption—that is, one could not "close" a role. This introduces some difficulties, as shown by the following modified specification for a (totally involved) gardener:

(cdef TI-GARDENER (and PERSON (all Hobby GARDENING-ACTIVITY)))

Given (PERSON *p*), (Hobby *p a*) and (GARDENING-ACTIVITY *a*), PENNI could not conclude that *p* was a [TI-GARDENER], because there might be other [Hobby]'s for *p* that were not [GARDENING-ACTIVITY]'s. However, for roles that are single-valued, PENNI could draw the correct conclusion. In other words, if [PERSON] included the specification (atmost 1 Hobby), PENNI would then be able to conclude (TI-GARDENER *p*).

A further consideration was that of counting. As mentioned earlier, RUP (and thus PENNI) allowed assertions of equality between individual constants, which was considered essential to a number of applications. However, as a consequence, one could not determine that distinct constants denoted distinct individuals unless it was asserted specifically. For example, given (Child *Joe Mary*) and (Child *Joe Pete*), one could not determine that *Joe* had at least two children unless one also knew that $Pete \neq Mary$. Of course, these assertions could be made, and, in general, applications did so. To simplify such assertions, PENNI introduced a special declaration called LPN where (LPN *a*) asserted that the constant *a* was guaranteed to denote an individual that was distinct from that denoted by any other constant. Like all assertions in PENNI, use of LPN was retractable.

Overall, NIKL offered a fairly wide expressive power, going beyond that of KL-ONE. KL-TWO built upon that power and offered a fairly powerful hybrid system that could accept, manipulate and reason about assertions. Both NIKL and KL-TWO found their way into a number of applications.

5.2.2. KNET and KloneTalk

Early in the 1980's, and prior to NIKL, a number of other KL-ONE-like systems were developed. Two of these were KNET [74–76], which was written in Prolog, and KloneTalk [70,71], which was written in SmallTalk. Both included the basic notions from KL-ONE plus some extensions. In the KNET effort at Burroughs/UNISYS and Harvard University, Freeman and Leitner examined *qua* concepts and extended structural descriptions. A qua concept is a concept representing the filler of a role in a certain context. For example, a tenant is a person who fills the renter role in a leasing-agreement. A more elaborate qua concept would show other connections, such as that each tenant has a landlord who happens to be the person who fills the lessor role in that same leasing-agreement. Freeman and Leitner's work on structural descriptions put those constructs on a more firm foundation (note that this work pre-dates the efforts to give semantic accounts) and showed additional inferences that could be drawn from them. In a similar fashion, the work on KloneTalk at Xerox PARC included a limited form of qua concept and elaborated the types of conclusions one could reach regarding structural descriptions and role value maps (again, this was done before formal semantic accounts were common).

A simple version of qua concepts found its way indirectly into NIKL—simple qua concepts were expressible using atleast, and-role, inv and range. For example, tenant could be described with the following:

(cdef TENANT (atleast 1 (and-role (inv Renter) (range LEASING-AGREEMENT))))

Regarding (1), if PENNI is told that (GARDENER *g*), it would then respond with "true" when asked the truth status of (PERSON *g*). Regarding (2), if instead PENNI is told that (PERSON *p*), (Hobby *p a*) and (GARDENING-ACTIVITY *a*) are all true, it would then respond with "true" when asked the truth status of (GARDENER *p*).

While the initial motivation for KL-TWO arose from a need to expand NIKL, the designers soon realized that the A-box would receive primary use in many applications, rendering the T-box as secondary. The hard part in designing KL-TWO was getting the TMS to work correctly with the classifier. The TMS is designed to draw incremental inferences and to record each one. The classifier, on the other hand, is designed to draw larger scale inferences and not to record them. Bridging these differences so that the TMS worked correctly took considerable effort.

PENNI always made the open world assumption—that is, one could not "close" a role. This introduces some difficulties, as shown by the following modified specification for a (totally involved) gardener:

(cdef TI-GARDENER (and PERSON (all Hobby GARDENING-ACTIVITY)))

Given (PERSON *p*), (Hobby *p a*) and (GARDENING-ACTIVITY *a*), PENNI could not conclude that *p* was a [TI-GARDENER], because there might be other [Hobby]'s for *p* that were not [GARDENING-ACTIVITY]'s. However, for roles that are single-valued, PENNI could draw the correct conclusion. In other words, if [PERSON] included the specification (atmost 1 Hobby), PENNI would then be able to conclude (TI-GARDENER *p*).

A further consideration was that of counting. As mentioned earlier, RUP (and thus PENNI) allowed assertions of equality between individual constants, which was considered essential to a number of applications. However, as a consequence, one could not determine that distinct constants denoted distinct individuals unless it was asserted specifically. For example, given (Child *Joe Mary*) and (Child *Joe Pete*), one could not determine that *Joe* had at least two children unless one also knew that $Pete \neq Mary$. Of course, these assertions could be made, and, in general, applications did so. To simplify such assertions, PENNI introduced a special declaration called LPN where (LPN *a*) asserted that the constant *a* was guaranteed to denote an individual that was distinct from that denoted by any other constant. Like all assertions in PENNI, use of LPN was retractable.

Overall, NIKL offered a fairly wide expressive power, going beyond that of KL-ONE. KL-TWO built upon that power and offered a fairly powerful hybrid system that could accept, manipulate and reason about assertions. Both NIKL and KL-TWO found their way into a number of applications.

5.2.2. KNET and KloneTalk

Early in the 1980's, and prior to NIKL, a number of other KL-ONE-like systems were developed. Two of these were KNET [74-76], which was written in Prolog, and KloneTalk [70,71], which was written in SmallTalk. Both included the basic notions from KL-ONE plus some extensions. In the KNET effort at Burroughs/UNISYS and Harvard University, Freeman and Leitner examined *qua* concepts and extended structural descriptions. A qua concept is a concept representing the filler of a role in a certain context. For example, a tenant is a person who fills the renter role in a leasing-agreement. A more elaborate qua concept would show other connections, such as that each tenant has a landlord who happens to be the person who fills the lessor role in that same leasing-agreement. Freeman and Leitner's work on structural descriptions put those constructs on a more firm foundation (note that this work pre-dates the efforts to give semantic accounts) and showed additional inferences that could be drawn from them. In a similar fashion, the work on KloneTalk at Xerox PARC included a limited form of qua concept and elaborated the types of conclusions one could reach regarding structural descriptions and role value maps (again, this was done before formal semantic accounts were common).

A simple version of qua concepts found its way indirectly into NIKL—simple qua concepts were expressible using atleast, and-role, inv and range. For example, tenant could be described with the following:

(cdef TENANT (atleast 1 (and-role (inv Renter) (range LEASING-AGREEMENT))))

KloneTalk introduced inheritance along the rvm chain in its completion algorithm and strengthened the inferences that could be drawn about structural descriptions and role value maps. These techniques were subsequently incorporated into NIKL's completion algorithm and many subsequent systems.

5.2.3. LOOM

LOOM [16,82,121] is the successor to NIKL and is being developed at USC/ISI. LOOM's goal is to provide a near-total representational and reasoning environment in which users can express and program nearly all they need for their applications.

LOOM's T-box language is the most expansive of all the systems we will survey. It includes all of $\mathcal{KL}$ *except* sd, or-role, not-role and disjoint-role. In addition, it allows representation of finitely enumerated sets, both ordered and unordered, as concepts. It allows representation of numeric intervals, including the union of multiple intervals, as concepts. Finally, it allows the representation of rules for classifying instances of one concept under another, similar to those we described earlier for CLASSIC. (As we mentioned before, the CONSUL effort pioneered the development of such rules, and LOOM was the first terminological KR system to incorporate such rules as part of the KR system.) The A-box language for LOOM is capable of recording information equivalent to a standard data base system—that is information equivalent to unnegated literals in the first order predicate calculus. Its query language, on the other hand, supports queries with the expressive power of the full first order predicate calculus—i.e., including conjunction, disjunction, negation, and quantification.

LOOM's services include subsumption, realization, truth maintenance (for both the T and A boxes), default reasoning, production rules, pattern-driven procedures, a pattern classifier (for aiding conflict resolution), and automatic detection of inconsistencies. The classifier has some ability to classify first-order expressions consisting of conjunctions of literals with variables. Moreover, LOOM attempts to integrate at least four types of programming common to AI: object-oriented programming (message passing), data-driven programming (production rules), problem solving (as in SOAR [122]), and constraint programming. LOOM thus occupies the opposite end of the spectrum from systems with small languages and limited services.

The reasons for LOOM's approach are several and are articulated in [16]. In summary, LOOM's designers note that the inferential operations needed by many applications are intractable in general. The KR designer must thus choose whether or not these needs are expressed inside or outside of the representation system. Systems such as KRYPTON place them outside the KR "black box," leaving a clean and predictable KR system but requiring the user to provide what's missing. LOOM provides facilities to express such needs within the KR system, where the resulting difficulties are dealt with by the application programmer. The advantage from LOOM's perspective is that the user is offered a consistent representation scheme—there is no need to construct ad hoc supplements to the KR system.

The final story on LOOM is not available as the system is still under development. However, it is already in use by several applications.

5.3. Other KL-ONE-like Systems

While not following directly in the paths of either KRYPTON nor KL-ONE, a number of systems have been developed that were strongly influenced by both. We now examine several of these systems, namely, BACK, MESON and SPHINX.

5.3.1. BACK

A separate effort begun in the mid-1980's was the development of BACK [54–57] at the Technical University of Berlin. The principal design motivation behind BACK was to construct a tightly-coupled hybrid system with balanced expressiveness. The notions of *tightly coupled* and *balanced expressiveness* are in contrast to the design of KL-TWO, which we discussed earlier. Basically, the BACK designers tried to offer complementary expressiveness in the T and A boxes.

Tractable and complete subsumption was too restrictive, so they did not require it. However, they wanted terminological reasoning to be manageable, so they did not offer wide expressibility.

BACK's T-box language included and, all, atleast, atmost, rvm=, individual, disjoint, cprim and cdef for concepts and and-role, domain, range, rprim and rdef for roles. BACK also included a method for introducing a term that was defined by an explicit, extensionally defined set. Subsumption in BACK is intractable [115], so a tractable but incomplete algorithm was included. Disjointness was computed as well.

BACK's A-box language allowed for the basic database operations. However, a "value" for a role filler could be a single constant, a set of constants, or a disjunction of sets of constants. Furthermore, a value could be a cardinality restriction on the number of distinct role fillers. This number restriction, expressed with card(*min*, *max*), identified the minimum and maximum number of fillers and thus allowed a local declaration of the closed world assumption, which BACK took advantage of. As a result, the A-box allowed for local forms of incomplete information plus a limited quantification using card. The main inferences that the A-box tried to draw were realization, consistency and whether one A-box expression was more general than another.

The balance in terms of expressiveness was achieved in several ways. Terms in the T-box could be used to assert information about individuals in the A-box. Moreover, the A-box utilized the restrictions expressed about such terms in the T-box (of course, all KL-ONE-like hybrid systems did this as well). Beyond that, distinct constants in BACK's A-box are assumed to denote distinct objects so that the number of role fillers could be counted and thus compared against number restrictions (KRYPTON was the first to do this). Also, with the card operator, it is possible to close a role for a given individual, i.e., to assert that the system knows all there is to know about it. Overall, BACK was successful in meeting its stated goals and found its way into a number of applications (e.g., [123]).

5.3.2. MESON

The MESON system [86,87], developed at Philips in Hamburg, Germany, beginning in the mid-1980's, aimed at a unified view of knowledge- and data-base management systems. Its goal was to combine the ability of database systems to handle large numbers of facts with that of terminological knowledge base systems to handle and reason about complex types. The language for describing terms was compact. The language for making assertions mainly allowed simple database operations.

MESON's term language included and, all, atleast, atmost, disjoint, cprim and cdef for concepts and no operators for roles. It also offered an operator for specifying implication rules similar to that of LOOM's, namely, $A \Rightarrow B$ where A and B are concepts. The A-box offered several database-like assertion and retrieval operations, which we will not examine here. In [86], MESON's developers present their system in terms of a formal data model in the spirit of [124]. Finally, MESON has been used in several applications, most notably for a configuration application [87].

5.3.3. SPHINX

The SPHINX system [94–96] was modeled after KL-TWO but with an assertional component based on Horn clauses as opposed to KL-TWO's propositional logic. SPHINX is basically a Prolog-based theorem prover for Horn clauses that is augmented with a classification-based reasoner.

The term language includes and, all, some, c-some, cprim and cdef for concepts and range, rprim and rdef for roles. It also allows a concept to be partitioned into a set of other concepts, thus combining the or and disjoint operators. It offers algorithms to determine subsumption and disjointness, of which the subsumption algorithm is incomplete (the problem turns out to be NP-hard for SPHINX's term language).

The language for making assertions is that of Horn clauses where the terms in the T-box are available for use. The language for queries is an extended language that includes certain logical connectives and quantifiers. The system takes a query and tries to prove it true, as in Prolog, using all information in both the T and A boxes. The T-box information is represented in the A-box as deduction rules, which are generated automatically by the T-box. SPHINX provides a simple truth-maintenance system and allows retraction of simple assertions. It does not have explicit negation, but uses negation-as-failure instead, again as in Prolog. It thus makes a closed world assumption.

5.3.4. Other Systems

In addition to the systems discussed above, there are many other systems that have pursued variations on the KL-ONE themes, some of them directly influenced by KL-ONE, and some of them independently developing similar or related themes. In the next few paragraphs we will describe a sampling of such systems and related work to give some idea of the diversity of approaches and issues. It turns out that there are a great many more works than we could possibly include.

Systems Oriented Toward Graphical Interfaces

KL-CONC and KL-MAGMA [66–69], developed by Cappelli and Moretti and others at CNR in Pisa, are implementations of KL-ONE and Brachman's SI-NETS. These systems focus on mechanisms for user interaction with the knowledge base—e.g., through Macintosh windows. KL-MAGMA runs on a variety of machines, including Macintosh, IBM-PC, VAX and SUN, and IBM mainframes.

SB-ONE [92,93] concentrates on the issues of providing an integrated environment for a user to interact with the knowledge representation structures. It was developed in the context of the XTRA natural language system, and its goals include the extension of KL-ONE style representations to handle sets and to support their use in a natural language processing [93]. Its developers stress the importance of the graphical interface for showing relationships between roles and concepts, not just displaying separate role and concept hierarchies.

A direct descendant of NIKL was KREME [77], developed at BBN in the late 1980's. KREME began as a copy of NIKL with its focus placed on the user interface. The result was a system that made it much easier to enter and edit NIKL taxonomies. Eventually, KREME added a few constructs of its own (e.g., a simple mechanism for supplying default values), but its main contribution was its very usable interface.

Database-Oriented Systems

K-REP [78] is a system developed at IBM in Yorktown Heights, New York, for database applications. Unlike most KR research, it has addressed database issues such as sharing and persistent storage objects. Its central application is a financial marketing expert system for assisting the design of financial solutions for the acquisition of large computer systems.

CANDIDE [58] is a database system that provides a T-box-like language for describing database types and uses classification as the basis for query processing.

Alternative Semantic Accounts

DRL [61–64] is a knowledge representation language conceived by Guarino as a revisitation of hybrid systems like KRYPTON within the framework of many-sorted logics. He argues that terminological knowledge needs a true intensional semantics (free of existential commitments), which is not the case for current many-sorted systems.

Another type of alternative semantic account is that of Patel-Schneider's, which we have already discussed in Section 5.1.3. and will discuss further in 5.4.

Systems Addressing Related Issues

In addition to systems that are directly descended from KL-ONE or derived from it, there are a number of other systems that have developed similar or related themes. For example, OMEGA

[125,126] is a description-oriented logic originally developed at MIT and continued under the European ESPRIT project that has been used in the context of office automation. It offers a higher order logic along with a formal account of inheritance and attributions and has a sound and complete axiomatization. Taxonomic reasoning on a lattice of descriptions is combined with deduction strategies defined at the metalevel.

CAKE[104–106] is a many layered hybrid representation system developed at MIT. It has formed the backbone of the Programmer's Apprentice project [127,128], which studies automatic programming. CAKE performs quick and shallow deductions automatically, and supports forward and backward chaining. At its base is a truth-maintenance system with equality called BREAD, which is built on RUP[119,120]. On top of BREAD is a layer, called FRAPPE, that implements a typed logic with special purpose procedures for sets, partial functions and frames, and for various algebraic properties of operators (e.g., commutativity and associativity). On the top layer rests the domain specific implementation of the Plan Calculus, used for reasoning about programs.

David McAllester, Bob Givan and T. Fatima, also at MIT, have developed a taxonomic syntax for first order inference [129] that uses quantifier-free taxonomic literals that they prove are more expressive than literals of first order logic and have a polynomial time decision procedure for satisfiability.

Hassan Aït-Kaci [130] working at the University of Pennsylvania developed a taxonomic system that is intended to be a structured type theory for a programming language—specifically, for a logic programming language whose computations are defined by "type checking." His work is well grounded in algebra and lattice theory and has been pursued and extended at a number of subsequent institutions [131,132].

L-LILOG [81], developed at IBM in Stuttgart, Germany, has produced a series of results dealing with reasoning using attributive concept descriptions and with feature logics and unification. With its primary application in natural language processing, L-LILOG integrates frame-like feature-value descriptions used in computational linguistics into an order-sorted predicate logic framework.

5.4. *The Complexity of Subsumption*

In [98], Brachman and Levesque raised the issue of the complexity of subsumption and pointed out an unexpected "computational cliff." Subsumption in the language they called $\mathcal{FL}^-$, consisting of the concept operators and, all and some and no role operators, took polynomial time with respect to the size of the terms in the worst case. Subsumption in $\mathcal{FL}^-$ was thus tractable. However, by adding the role operator restr (which they called *VRdiff*), subsumption became co-NP-hard and thus intractable. Their conclusion was that very subtle changes in the term language could lead to dramatic differences in the complexity of subsumption.

This effort launched a series of studies into various term languages and the resulting complexity of subsumption and coherence. Table 2 presents the highlights of these studies.[22] As can be seen, there are several of these "cliffs." For example, subsumption in MESON, with concept operators and, all, atleast and atmost, has polynomial worst case behavior. By changing atleast to c-atleast, one gets KANDOR, in which subsumption is co-NP-hard.

Another important and surprising "cliff" is when subsumption moves from being a decidable problem to an undecidable one. Patel-Schneider showed that subsumption in NIKL is undecidable [133]. Schmidt-Schauß introduced a minimal language $\mathcal{ALR}$, which includes only the concept operators and, all and rvm= with no role operators, and he showed that subsumption in $\mathcal{ALR}$ is still undecidable [134]! Obviously, $\mathcal{ALR}$ is a very small subset of KL-ONE and NIKL. Clearly the rvm= operator, almost by itself, introduces considerable complexity. Schmidt-Schauß also shows, however, that if all roles are functional (i.e., single valued), then subsumption becomes tractable [134].[23]

[22]Some of the information in Table 2 was gathered from a similar table distributed by Bernhard Nebel at the Workshop on Term Subsumption Languages in Knowledge Representation, Jackson, NH, October, 1989.

[23]We note that some of these results came out of research that connects unification grammars with terminological KR systems (e.g., [135,136]). In unification grammar terminology, rvm= is called the *agreement* operator and functional roles are called *features*.

Table 2. Summary of worst case complexity results.

Name	Concept Ops.	Role Ops.	Subsumption	Coherence
$\mathcal{FL}^-$	and, all, some		polynomial (A)	all concepts coherent(B)
$\mathcal{FL}$	and, all, some	restr	co-NP-hard (A)	all concepts coherent(B)
KRYPTON	and, all	chain	polynomial (C)	all concepts coherent
MESON	and, all, atleast, atmost		polynomial	
KANDOR	and, all, c-atleast, atmost		co-NP-hard (D)	
BACK	and, all, atleast, atmost	and-role	co-NP-hard (D)	
$\mathcal{AL}$	and, simple-not, all, some			linear (B)
$\mathcal{ALE}$	and, simple-not, all, c-some		NP-complete (E)	co-NP-hard (E)
$\mathcal{ALU}$	and, or, simple-not, all, some		co-NP-hard (B)	NP-complete (B)
$\mathcal{ALC}$	and, or, not, all, c-some		P-SPACE-complete(B)	P-SPACE-complete(B)
$\mathcal{ALC}$ w/ number constr.	and, or, not, all, c-some, atleast, atmost		P-SPACE-complete(F)	P-SPACE-complete(F)
$\mathcal{ALC}$ w/ role hier.	and, or, not, all, c-some	and-role	P-SPACE-complete(F)	P-SPACE-complete(F)
$\mathcal{ALC}$ w/ both number constr. & role hier.	and, or, not, all, c-some, atleast, atmost	and-role	decidable (F)	decidable (F)
$\mathcal{ALCA}$	and, or, not, all, c-some, rvm=		undecidable (G)	undecidable (G)
$\mathcal{ALCA}$ w/ only functional roles	and, or, not, all, c-some, f-rvm=		co-NP-complete (H)	NP-complete (H)
$\mathcal{ALCA}$ where rvm= used only w/ functional roles	and, or, not, all, c-some, f-rvm=		decidable (F)	
$\mathcal{ALR}$	and, all, rvm=		undecidable (G)	
NIKL	and, all, atleast, atmost, rvm, sd, disjoint	and-role, domain range, inv	undecidable (G,I)	

The above complexity results are worst case in time with respect to the size of the terms being compared. All results assume there is *no* separate terminology (i.e., all assume that cprim, cdef, rprim and rdef are not allowed). A blank box means that the complexity is unknown. A missing citation means that the result is informal. simple-not is a restricted not operator that can apply only to atomic concepts. f-rvm= is a restricted rvm= operator that can apply only to functional roles (also called "features"). Citations for the above results follow.

A: [98]	D: [115]	G: [134]
B: [103]	E: [137]	H: [135,139]
C: [97]	F: [138]	I: [133]

There is another surprising source of complexity. All of the foregoing results do not incorporate an actual terminology. In other words, they assume that all names of concepts or roles that appear within the terms are primitive and atomic (i.e., do not have definitions). When one incorporates a terminology (in our case, using cprim, cdef, rprim and/or rdef), then another source of complexity arises. Nebel has shown that the problem of classification in such cases is co-NP-complete [140]. Thus, while subsumption between two fully expanded terms in, say, KRYPTON is polynomial, the overall subsumption complexity, if it includes expanding the terminological definitions, is co-NP-complete. Moreover, he states that we cannot escape this worst-case complexity even if we assume that all subsumption relations between previously defined terms are already known and the subsumption algorithm takes advantage of that information. Thus, even in the simplest of T-box languages, worst-case subsumption is intractable if term definitions are allowed. Fortunately, as Nebel points out, the performance of classifiers on actual terminologies found in applications is frequently, if not consistently, tractable. Apparently, the cases that lead to combinatorial explosions either do not arise or arise very infrequently. Thus, the practical effect of this result may not be overly negative.

We conclude this section by examining an alternative approach to studying complexity. In [141], Patel-Schneider offers a method to describe the inferences drawn by classifiers similar to that of NIKL by introducing an alternative style of semantics (this was briefly discussed in Section 5.1.3., where we examined MANTRA). The NIKL classifier is incomplete with respect to the semantics of NIKL. Several unsuccessful efforts were made by NIKL's developers to describe succinctly the class of subsumptions actually discovered by NIKL's classifier. Patel-Schneider successfully produced an approximation to such a description using the following approach. He introduced a language similar to that of NIKL and provided an alternative semantics that was *weak* in the sense that it was capable of supporting far fewer inferences than would be supported using the standard account. In particular, this alternative account allows no chaining of results—e.g., it does not support modus ponens, the rule of classical logic that combines an if-then rule with the truth of its antecedent to conclude the consequent. The semantics he provided was based on a four-valued relevance logic that yields a weak entailment relationship. Patel-Schneider argues that the subsumptions supported by his semantics is very close to the set of subsumptions that the NIKL classifier actually determines. Thus, this account closely approximates the sought-after account, although at the cost of introducing a nontraditional semantic account.

5.5. *Critiques and Dissenting Opinions*

While KL-ONE and its successors have been used for a variety of applications and have contributed many useful capabilities, not all users have been completely satisfied. (The general tenor of the objections, however, are in the direction of "we want more" rather than "we don't want this.") An extensive critique of NIKL is offered by Haimowitz [142], much of which applies to nearly all the systems reviewed in this paper. Haimowitz was part of the medical diagnosis research group at Massachusetts Institute of Technology (MIT) that developed, among other systems, ABEL, a medical diagnostic program for acid-base and electrolyte disorders. That group used NIKL for representing part of their medical knowledge base and found that the basic taxonomic functionality was very useful for certain aspects of medical knowledge. However, NIKL was not able to make the desired inferences regarding other aspects, such as with regards to transitive relations (e.g., part-whole), symmetric relations (e.g., connected-to), causality, intervals and sequences. Also, NIKL did not provide a way of specifying sufficient conditions for recognition, since the specification of a term in NIKL must be either necessary or both necessary and sufficient. Overall, the comments in [142] and another critique by Smoliar and Swartout [143], which addresses the limitations of the NIKL classifier in the context of the Explainable Expert System (EES) project [9,10], provide a good review of NIKL's uses and limitations, and they identify topics for future research. An analysis of these comments is made in [91]. Woods [28] offers somewhat different criticisms and proposes methods to overcome some of the deficiencies.

Some researchers take issue with some of the design goals of Brachman and Levesque [98,113] that have affected many the works reviewed in this section. Most vocal are Doyle and Patil [144], who argue against limiting expressiveness in order to guarantee complete and tractable inferential algorithms. The net effect, they claim, is a language so severely limited that it is no longer of general use, even though it may find specific use in some applications. They also argue against the restriction that the classifier operate only with respect to purely definitional information. As an alternative, they urge that KR languages offer fully expressive languages and that classification take contingent information into account. They argue further that completeness and tractability are poor measures of a KR system's utility, and that broader notions of utility and rationality as found in decision theory should be used instead. A final suggestion is that a KR system should offer inference tools along with ways to manage them, should support approximate (and possibly unsound) forms of recognition, and should allow classification of definitions involving defaults.

6. SUMMARY, CONCLUSIONS, AND FUTURE DIRECTIONS

KL-ONE marked a transition from knowledge representation systems that were essentially ad hoc data structures for managing certain kinds of computation, to systems with an external semantic criterion to which both the representation of knowledge and the algorithms that operate on that knowledge should both be faithful. As such, it generated tremendous interest, both in the formal community and in a large community of potential users. As in most pioneering systems, the original KL-ONE system did not fully achieve all of its goals. However, it has pointed the way to an active research area that is still making advances and has a great deal of future promise.

An important goal of KL-ONE was to make useful KR services available to a wide audience. Thus, expressive power in early KL-ONE increased as the system developed. This expansion was limited, however, by a number of factors so that the resulting language remained less expressive in most respects than, say, first-order predicate calculus. Some of the limitations came from the intuitions of the developers along the lines of "what *should and should not* be included in a terminological language."[24] Default information, for example, was expressly prohibited from KL-ONE since defaults are not necessary conditions. Other limitations were motivated by concerns about the system's ability to draw conclusions for unrestricted kinds of inference. A direct representation of sequences, for example, was held off, in part, because the KL-ONE developers were unclear about how well and how easily the classifier could determine subsumption using them. The overall direction, however, was toward providing more, not less, expressive power.

As we have seen, this goal of increasing expressiveness took several turns in subsequent research, resulting in substantial decreases in expressive power in many systems. For some users, these simpler systems that offered less expressive power were exactly what was needed. For a number of other users, however, even the more expressive systems were inadequate (e.g., [16,102]).

One of the areas where KL-ONE both succeeded and failed is in the distinction between definitional and assertional information. As we have seen, this distinction has become a fundamental principle in much of the research that has followed KL-ONE, especially in KRYPTON and its successors. However, in much of this research, a number of distinctions related to this one, but not quite the same, have become aligned as if they were the same distinction. For example, conceptual structure is often equated with definition; definition is equated with necessary and sufficient conditions; and necessary conditions that are not sufficient are equated with assertional information. Similarly, structured subsumption is equated with terminological knowledge; and terminological knowledge is equated with efficient taxonomic operations. One result of all this is that users of these systems often cannot understand some of the subtle distinctions made by the developers. As MacGregor puts it, "Our experience with NIKL and LOOM suggests that drawing such a distinction [between terminological and assertional knowledge] confuses (all but the most sophisticated) users as often as it helps them" [16].

This situation is exacerbated by the fact that these systems (especially the more limited ones) often don't provide some of the capabilities that some users need. When this happens, users get creative and the intended semantics of the notation is easily cast aside for an operational semantics determined by what the algorithms do (to the extent that they can be understood). In many cases, users have creatively used these systems to achieve behavior that was not intended by the designers. For example, the USC/ISI CONSUL group ignored the provisions in KL-ONE for individual concepts because they wanted to further specialize individual descriptions, an operation that KL-ONE did not permit; they used generic descriptions instead. In a similar way, many users have wanted to use the classification operation to take into account information that is assertional rather than terminological or definitional. While a few systems permit this, many of the others can be manipulated to achieve that effect by ignoring the intended semantics of the notations. There are some lessons here to be digested. Woods [28] proposes a framework that may provide a solution. For example, he points out that assertional and definitional functionality are not necessarily mutually exclusive and that some notational devices may simultaneously have both aspects.

At this point in time, the surfeit of intractability results seems to have reached its logical end with the conclusion that practically everything of any use is intractable (in the worst case). The research mood is shifting to favor incomplete systems with increased expressivity and tractable

[24]We are not being temporally accurate in our use of terminology here, since at the time of KL-ONE's development, the word "terminological" was not in common use. However, its common usage nowadays provides a convenient hook to identify the KL-ONE line of research.

algorithms. However, there is a need for new, more realistic goals to replace deductive completeness and worst case tractability. We need new ways to characterize the class of inferences that a taxonomic network can perform and to more realistically characterize the time required to perform them. For example, Woods [28] proposes a definition of "intensional" subsumption, that involves an almost psychological distinction between the kinds of subsumption that people (and systems) should be able to do rapidly and those that require more complex deduction. Woods argues that complexity with respect to knowledge base size is a better measure of utility than complexity with respect to the size of individual conceptual descriptions, and he shows that a version of intensional subsumption has a typical case complexity that is sublinear in the size of the knowledge base.

While redirecting the focus of formal analysis of taxonomic systems is a useful direction for further research, there are other research directions in need of attention as well. For example, we mentioned earlier the relative neglect of issues such as "individuals" that have properties but either do not exist or are not distinct from other individuals known to the system. Such things do not fit well in a first-order semantics, yet almost all formal work in this area has been done in the first-order context. More work devoted to understanding intensional entities that cannot be adequately characterized with a first-order semantics is definitely in order. Such structures are essential in natural language applications involving belief modeling (e.g., [145]) and intentional plan recognition (e.g., [146]).

Finally, we need to devote more attention to discovering efficient algorithms for useful taxonomic operations and other related operations. For most purposes, it is not enough to know that an algorithm is polynomial or exponential time. We need to know how rapidly it can be done, and if there are intractable worst cases, we need to know efficient algorithms for useful subclasses of the problem. One obstacle to making advances in this area is the difficulty of gaining a sufficiently crisp statement of the problems such systems are expected to solve so that formal attention can be devoted to them. Woods [28] is one attempt to provide such a statement. Doyle and Patil [102] provide a substantial list of things that users would like a knowledge representation language to do. MacGregor [16] provides an implementor's perspective on what a knowledge representation should provide. There is much food for thought here, and a great deal of room for further progress.

REFERENCES

1. W.A. Woods, What's in a link? Foundations for semantic networks. In *Representation and Understanding: Studies in Cognitive Science*, (D.G. Bobrow and A. Collins, Eds.), pp. 35–82, Academic Press, New York, (1975). Reprinted in *Readings in Knowledge Representation*, (Edited by R. Brachman and H. Levesque), Morgan-Kaufman, San Mateo, CA, 1985, and in *Readings in Cognitive Science*, (Edited by A. Collins and E.E. Smith), Morgan-Kaufman, San Mateo, CA, (1988).
2. R.J. Brachman, A structural paradigm for representing knowledge, Ph.D. Thesis, Harvard Univ., (1977), also, BBN Report No. 3605, Bolt Beranek and Newman Inc., (May, 1978).
3. W.A. Woods and R.J. Brachman, Research in natural language understanding, Quarterly Technical Progress Report No. 1: 1 Sep 77 - 30 Nov 77, BBN Report No. 3742, Bolt Beranek and Newman Inc., (December, 1977).
4. R.J. Brachman and J.G. Schmolze, An overview of the KL-ONE knowledge representation system, *Cognitive Science* 9(2), 171–216, (April–June, 1985).
5. W. Mark, Representation and inference in the consul system, In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, Vancouver, BC, (August, 1981).
6. W. Mark, Natural-language help in the consul system, In *AFIPS Conference Proceedings*, (H.L. Morgan, Editor), 475–479, National Computer Conference, AFIPS Press, (June, 1982).
7. W. Mark, Use of database organization in the consul system, In *Proceedings of the Workshop on Data Abstraction, Databases, and Conceptual Modelling*, Joint SIGART/SIGPLAN/SIGMOD (ACM) Publication, (February, 1981).
8. D. Wilczynski, Knowledge acquisition in the consul system, In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, (1981).
9. R. Neches, W.R. Swartout and J.D. Moore, Enhanced maintenance and explanation of expert systems through explicit models of their development, *IEEE Transactions on Software Engineering* SE-11(11), 1337–1351, (November, 1985).

10. W.R. Swartout and S.W. Smoliar, On making expert systems more like experts, *Expert Systems* **4**(3), 196–207, (August, 1987).

11. J. Bateman, R. Kasper, J. Schütz and E. Steiner, A new view of the process of translation, In *Proceedings of EACL-89, the 4th Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, England, (June, 1989).

12. R. Kasper, An experimental parser for systemic grammars, In *Proceedings of Coling-88, the 12th International Conference on Computational Linguistics*, Budapest, Hungary, (August, 1988).

13. R. Kasper, Unification and classification: An experiment in information-based parsing, In *Proceedings of the International Workshop on Parsing Technologies*, Pittsburgh, PA, (August, 1989).

14. Y. Arens, L. Miller, S.C. Shapiro and N.K. Sondheimer, Automatic construction of user-interface displays, In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, 808–813, St. Paul, Minn., (August, 1988).

15. J.G. Schmolze and R.J. Brachman, Proceedings of the 1981 KL-ONE workshop, Technical Report 4842, Bolt Beranek and Newman Inc., (June 1982). Also appears as Fairchild Technical Report No. 618, (May, 1982).

16. R. MacGregor, The evolving technology of classification-based knowledge representation systems, In *Principles of semantic networks: Explorations in the representation of knowledge*, (J. Sowa, Editor), Morgan-Kaufman, San Mateo, CA, (1991).

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

18. Fahlman, *NETL: A System For Representing and Using Real-World Knowledge*, The M.I.T. Press, (1979).

19. W.A. Woods, Don't blame the tool, *Computational Intelligence* **3**(3), 228–237, (August, 1987).

20. W.A. Woods, Important issues in knowledge representation, *Proceedings of the IEEE* **74**(10), (October, 1986).

21. W.A. Woods, Parallel algorithms for real time knowledge based systems, In *Research in Natural Language Understanding: Quarterly Progress Report No. 6, December 1, 1978–February 28, 1979*, (Edited by W.A. Woods), Technical Report 4181, (April, 1979).

22. W.A. Woods, The JARGON language, In *Theoretical Studies in Natural Language Understanding, Annual Report, May 1, 1978–April 30, 1979*, (Edited by W.A. Woods), BBN Report 4332, Cambridge, MA, (April, 1979).

23. R.J. Brachman, R.J. Bobrow, P.R. Cohen, J.W. Klovstad, B.L. Webber and W.A. Woods, Research in natural language understanding, Annual Report (Sept. 1, 1978–Aug. 31, 1979), Technical Report 4274, Bolt Beranek and Newman Inc., Cambridge, MA, (1979).

24. T. Lipkis and W. Mark, Consul Note 5, The consul classifier, Technical report, USC/Information Sciences Institute, Marina del Rey, CA, (1981).

25. T.A. Lipkis, A KL-ONE classifier, In *Proceedings of the 1981 KL-ONE Workshop*, (Edited by J.G. Schmolze and R.J. Brachman), pp. 128–145, Bolt Beranek and Newman Inc. Report No. 4842, (June, 1982).

26. J.G. Schmolze and T.A. Lipkis, Classification in the KL-ONE knowledge representation system, In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Karlsruhe, West Germany, (August, 1983).

27. R.J. Brachman, I lied about the trees, *The AI Magazine* **VI**(**3**), 80–93, (Fall, 1985).

28. W.A. Woods, Understanding subsumption and taxonomy: A framework for progress, In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, (John Sowa, Editor), Morgan-Kaufmann, San Mateo, CA, (1991).

29. R.J. Brachman, On the epistemological status of semantic networks, *Associative Networks: Representation and Use of Knowledge by Computers*, (Edited by N.V. Findler), pp. 3–50, Academic Press, New York, (1979).

30. A. Newell, The knowledge level, *Artificial Intelligence* **18**(1), 87–127, (1982).

31. N.R. Greenfeld and M.D. Yonke, AIPS: an information presentation system for decision makers, BBNREP 4228, BBN, (December, 1979).

32. M.D. Yonke and N.R. Greenfeld, AIPS: An Information Presentation System for Decision Makers, In *Proceedings of Thirteenth Hawaii International Conference on System Sciences*, Volume II, pp. 48–56, Univ. of Hawaii, January, 1980, Also reprinted in Data Base, V.12, 1980. A revised version of this paper is also available as BBN Report No. 4228, Bolt Beranek and Newman Inc., (December, 1979).

33. F. Zdybel, N.R. Greenfeld, M.D. Yonke and J. Gibbons, An information presentation system, In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pp. 978–984, Vancouver, B. C., (1981).

34. M. Bates, D. Stallard and M. Moser, The IRUS transportable natural language database interface, In *Expert Database Systems*, (Edited by L. Kerschberg), Cummings Publishing Company, Menlo Park, CA, (1985).

35. W.A. Woods, Transition network grammars for natural language analysis, *CACM* **13**(10), pp. 591–606, (October 1970), Reprinted in *Tutorials: Context-Directed Pattern Recognition and Machine Intelligence Techniques for Information Processing*, (Edited by Y. Pao and G. W. Ernest), IEEE Computer Society Press, Silver Spring, MD, (1982) and in *Readings in Natural Language Processing*, (Edited by B. Grosz, K. Spark Jones and B. L. Webber), Morgan-Kaufman, San Mateo, CA, (1986).

36. W.A. Woods, Natural language communication with machines: An ongoing goal, In *Artificial Intelligence Applications for Business*, (Edited by W. Reitman), pp. 195–209, Ablex Corporation, Norwood, NJ, (1984).

37. J. Yen, A principled approach to reasoning about the specificity of rules, In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, (July, 1990).

38. J. Yen, R. Neches and R. MacGregor, Using terminological models to enhance the rule-based paradigm, In *Proceedings of the Second International Symposium on Artificial Intelligence*, Monterrey, Mexico, (October, 1989).

39. J. Yen, Reasoning about specificity of patterns in term subsumption-based systems, Technical Report TAMU 90-003, Dept. of Computer Science, Texas A&M University, (February, 1990).

40. R.J. Brachman and H.J. Levesque, Assertions in KL-One, In *Proceedings of the 1981 KL-ONE Workshop*, (Edited by J.G. Schmolze and R.J. Blachman), pp. 8-17, Bolt Bernek and Newman Inc. Report No. 4842, (1982).

41. R.J. Brachman, R.E. Fikes and H.J. Levesque, KRYPTON: Integrating terminology and assertion, In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pp. 31–35, Washington, D.C., (August, 1983).

42. Y. Jang and R. Patil, KOLA: A knowledge organization language, In *13th Symposium on Computer Applications in Medical Care*, (Edited by Lawrence C. Kingsland III), IEEE Computer Society Press, (1989).

43. Y. Jang, KOLA: Knowledge organization language, Technical Report LCS TR-396, MIT Laboratory for Computer Science, (1988).

44. M. Derthick, Mundane reasoning by parallel constraint satisfaction, Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, September (1988), available as Technical Report CMU-CS-88-182, Department of Computer Science, Carnegie Mellon University.

45. J.G. Schmolze, Terminological knowledge representation systems supporting *N*-ary terms, In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR89)*, Toronto, Canada, (May, 1989).

46. T. Finin, R. Fritzson, R. McEntire, D. McKay and A. O'Hare, The intelligent system server: Delivering AI to complex systems, In *Proceedings of the First International Workshop on Tools for AI, Architectures, Languages and Algorithms*, Fairfax, VA, (October, 1989).

47. T. Finin and R. Fritzson, How to serve knowledge—notes on the design of a knowledge base server, In *AAAI Spring Symposium on Knowledge System Development Tools and Languages*, (March, 1989).

48. D. McKay, T. Finin and A. O'Hare, The intelligent database interface: Integrating AI and database systems, In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, (July, 1990).

49. E. Decio, P. Petrin and L. Spampinato, Pushing the terminological barrier, In *Proceedings of the Workshop on Inheritance Hierarchies in Knowledge Representation and Programming Languages*, Viareggio, Italy, (February, 1989).

50. H. Bergmann and M. Gerlach, QUIRK—Implementierung einer TBox zur Repraesentation begrifflichen Wissens. Technical Report WISBER Report M11, University of Hamburg, (June, 1987), (in German).

51. M. Poesio, The QUARK Reference Manual, Technical Report WISBER Report M22, University of Hamburg, (June, 1988).

52. M. Koubarakis, J. Mylopoulos, M. Stanley and A. Borgida, Telos: Features and formalization, Technical Report KRR-TR-89-4, Dept. of Computer Science, University of Toronto, (1989).

53. H. Trost and B. Pfahringer, VIE-KL: An experiment in hybrid knowledge representation, Technical Report TR-88-8, Oesterreichisches Forschungsinstitut fuer Artificial Intelligence, Wien, (1988).

54. K. von Luck, B. Nebel, C. Peltason and A. Schmiedel, The anatomy of the BACK system, KIT Report 41, Department of Computer Science, Technische Universität Berlin, Berlin, West Germany, (January, 1987).

55. B. Nebel and K. von Luck, Hybrid reasoning in BACK, In *Methodologies for Intelligent Systems*, (Edited by Z.W. Ras and L. Saitta) Vol. 3, pp. 260–269. North-Holland, Amsterdam, Holland, (1988).

56. B. Nebel and K. von Luck, Issues of integration and balancing in hybrid knowledge representation systems, In *GWAI-87. 11th German Workshop on Artificial Intelligence*, (Edited by K. Morik) pp. 114–123, Springer-Verlag, Berlin, Germany, (1987).

57. C. Peltason, A. Schmiedel, C. Kindermann and J. Quantz, The BACK system revisited, Technical Report KIT Report 75, Department of Computer Science, Technische Universitat, Berlin, (August, 1989).

58. H.W. Beck, S.K. Gala and S.B. Navathe, Classification as a query processing technique in the CANDIDE semantic data model, In *Proceedings of IEEE Conference on Data Engineering*, Los Angeles, CA, (February, 1989).

59. A. Borgida, R.J. Brachman, D.L. McGuinness and L.A. Resnick, CLASSIC: A structural data model for objects, In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, ACM, Portland, Oregon, (June, 1989).

60. R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick and A. Borgida, Living with CLASSIC: When and how to use a KL-ONE-like language, In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, (Edited by J. Sowa) Morgan-Kaufmann, San Mateo, CA, (1991).

61. N. Guarino, DRL: terminologic and relational knowledge in Prolog, In *Proceedings of 8th European Conference on Artificial Intelligence (ECAI-88) Muenchen*, (Edited by Y. Kodratoff), (August 1-5, 1988).

62. N. Guarino, Representing domain structure of many-sorted Prolog knowledge bases, In *Foundations of logic and functional programming*, (Edited by G. Levi, M. Boscarol and L. Aiello), Springer-Verlag, Berlin (1988).

63. N. Guarino, Attributes and extensional equivalence in DRL, In *Proceedings of the 3rd International Symposium on Methodologies for Intelligent Systems (ISMIS-88)Torino* (October 12–15, 1988), (Edited by B. Radig and L. Saitta), North-Holland (1988).

64. N. Guarino, Nature and structure of terminological knowledge: The DRL approach, In *Proceedings of the 1st Conference of the Italian Association for Artificial Intelligence (AI*IA),Trento*, (November, 1989).

65. P.F. Patel-Schneider, Small can be beautiful in knowledge representation, In *Proceedings IEEE Workshop on Principles of Knowledge-Based Systems*, pp. 559–565, Denver, Colorado, (December, 1984), IEEE Computer Society, Extended version available as AI Technical Report No. 37, Schlumberger Palo Alto Research, (October, 1984).

66. A. Cappelli, L. Moretti and C. Vinchesi, KL-CONC: A language for interacting with SI-Nets, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, (1983).

67. A. Cappelli and L. Moretti, An approach to natural language in the SI-Nets paradigm, In *Proceedings of First Conference of ACL-Europe*, Pisa, (1983).

68. A. Cappelli, G. Caracoglia and L. Moretti, A chunking mechanism for a knowledge representation system, *Cybernetics and Systems* **17**(4), 277–287, (1986).

69. G. Adorni, A. Cappelli, S. Gaglio and L. Moretti, Integrating logic programming and structured knowledge representation, *Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology* **4**(1), (1987).

70. R.R. Fikes and A. Henderson, On supporting the use of procedures in office work, In *Proceedings of the First National Conference on Artificial Intelligence (AAAI-80)*, Stanford, CA, (1980).

71. R.R. Fikes, Highlights from KloneTalk, In *Proceedings of the 1981 KL-ONE Workshop*, (Edited by J.G. Schmolze and R.J. Brachman), pp. 8–17, Bolt Beranek and Newman Inc., Report No. 4842, (1982).

72. M. Vilain, KL-TWO, A hybrid knowledge representation system, BBN Technical Report 5694, BBN Laboratories, (September, 1984).

73. M. Vilain, The restricted language architecture of a hybrid representation system, In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 547–551, Los Angeles, CA, (August, 1985).

74. M.W. Freeman and H.H. Leitner, KNET Extensions, *Journal of Computational Linguistics*, (July–September, 1981).

75. M.W. Freeman, L. Hirschman, D.P. McKay and M.S. Palmer, KNET: A logic-based associative network system, In *Presented at the third Knowledge Representation Workshop*, Santa Barbara, CA, (October, 1983).

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

77. G. Abrett and M.H. Burstein, The KREME knowledge editing environment, *International Journal of Man-Machine Studies* **27**(2), 103–126, (1987).

78. E. Mays, C. Apte, J. Griesmer and J. Kastner, Experience with K-Rep: An object-centered knowledge representation language, In *Proceedings of IEEE AI Application Conference*, San Diego, California, (March, 1988).

79. R.J. Brachman, R.E. Fikes and H.J. Levesque, KRYPTON: A functional approach to knowledge representation, *IEEE Computer* **16**(10), 67–73, (October, 1983), Also available as Fairchild Technical Report No. 639 or as FLAIR Technical Report No. 16.

80. R.J. Brachman, V.P. Gilbert and H.J. Levesque, An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON, In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 532–539, Los Angeles, CA, (August, 1985).

81. U. Pletat and V. Luck, Knowledge representation in LILOG, Technical Report IWBS Report No. 90, IBM, Germany, Scientific Center, Institute for Knowledge Based System, (November, 1989).

82. R.M. MacGregor, A deductive pattern matcher, In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pp. 403–408, St. Paul, Minn., (August, 1988).

83. R. MacGregor and J. Yen, The knowledge representation project at ISI, Technical Report Tech. Report RR-87-199, USC/ISI, (1987).

84. G. Bittencourt, A unified formalism for knowledge representation, In *Proceedings of the Third International Symposium on Methodologies for Intelligent Systems (ISMIS)* , pp. 111–121, Torino, Italy, (October, 1988).

85. G. Bittencourt, A hybrid system architecture and its semantics, In *Proceedings of the Fourth International Symposium on Methodologies for Intelligent Systems (ISMIS)* , Charlotte, N.C., (October, 1989).

86. J. Edelmann and B. Owsnicki, Data models in knowledge representation systems: A case study, In *GWAI-86 und 2, Österreichische Artificial-Intelligence-Tagung*, (Edited by C.R. Rollinger and W. Horn), pp. 69–74, Springer-Verlag, Berlin, Germany, (1986).

87. B. Owsnicki-Klewe, Configuration as a consistency maintenance task, In *GWAI-88. 12th German Workshop on Artificial Intelligence*, (Edited by W. Hoeppner), pp. 77–87, Springer-Verlag, Berlin, Germany, (1988).

88. M.G. Moser, An overview of NIKL, The new implementation of KL-ONE, In *Research in Knowledge Representation for Natural Language Understanding, Annual Report (1 Sept. 1982 - 31 Aug. 1983). BBN Report No. 5421*, (Edited by C. Sidner, M. Bates, R. Bobrow, B. Goodman, A. Haas, R. Ingria, D. Israel, D. McAllester, M. Moser, J. Schmolze and M. Vilain), pp. 7–26, Bolt Beranek and Newman Inc., Cambridge, MA, (1983).

89. G. Robins, The NIKL Manual, Technical Report, Univ. of Southern California Information Sciences Institute, Marina del Rey, CA, (1986), (draft).

90. T.S. Kaczmarek, R. Bates and G. Robins, Recent developments in NIKL, In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pp. 978–987, Philadelphia, PA, (August, 1986).

91. J.G. Schmolze and W.S. Mark, The NIKL experience, *Computational Intelligence*, **6**(1), (1991).

92. A. Kobsa, Utilizing knowledge: The components of the SB-ONE knowledge representation workbench, In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, (Edited by J. Sowa), Morgan-Kaufmann, San Mateo, CA, (1991).

93. J. Allgayer and C. Reddig, What KL-ONE lookalikes need to cope with natural language—scope and aspect of plural noun phrases, In *Sorts and Types in Artificial Intelligence*, (Edited by K.H. Blaesius, U. Hedstueck and C.R. Rollinger), Springer-Verlag, Berlin, Germany, (1990).

94. S. Han, A frame and horn clause-based hybrid knowledge representation, Ph.D. Thesis, Department of Computer Science, Korea Advanced Institute of Science and Technology, Seoul, Korea, (1989).

95. S. Han, D.W. Shin, Y. Kim, Y.P. Jun, S.R. Maeng and J.W. Cho, A logic programming approach to hybrid knowledge representation, *Applied Artificial Intelligence: An International Journal* **2**, 93–127, Hemisphere Publishing Corporation, (1988).

96. S. Han and J.W. Cho, Sphinx—a hybrid knowledge representation system, In *Proceedings of International Conference on Fifth Generation Computer Systems*, Tokyo, pp. 1211–1220, (November, 1988).

97. P.F. Patel-Schneider, Decidable, logic-based knowledge representation, Ph.D. Thesis, Univ. of Toronto, (1987), Also available as AI Technical Report No. 56, Schlumberger Palo Alto Research, (May, 1987)

98. R.J. Brachman and H.J. Levesque, The tractibility of subsumption in frame-based description languages, In *Proceedings of the Fourth National Conference on Artificial Intelligence* (AAAI-84), pp. 34–37, (August, 1984).

99. B. Nebel, On terminological cycles, Technical Report KIT - Report 58, Technische Universitat Berlin, Germany, (November, 1987).

100. B. Nebel, Terminological cycles: Semantics and computational properties, In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, (Edited by J. Sowa), Morgan-Kaufmann, San Mateo, CA, (1991).

101. R.J. Brachman and H. Levesque, Competence in knowledge representation, In *Proceedings of the Second National Conference on Artificial Intelligence, (AAAI-82)*, Pittsburgh, PA, (August, 1982).

102. J. Doyle and R.S. Patil, Language restrictions, taxonomic classifications and the utility of representation services, Technical Report MIT/LCS/TM-387, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, MA, (May, 1989).

103. M. Schmidt-Schauß and G.Smolka, Attributive concept descriptions with complements, *Artificial Intelligence*, **48**(1), (February, 1991); also available as IWBS Report 68, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, (June, 1989).

104. C. Rich, Knowledge representation languages and the predicate calculus: How to have your cake and eat it too, In *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*, pp. 193–196, (August, 1982).

105. C. Rich, The layered architecture of a system for reasoning about programs, In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 540–546, Los Angeles, CA, (August, 1985).

106. Y.A. Feldman and C. Rich, Bread, frappe and cake: The gourmet's guide to automated deduction, In *Proceedings of 5th Israeli Symposium on Artificial Intelligence*, Tel Aviv, Israel, (December, 1988).

107. P.J. Hayes, The Logic of frames, In *Frame Conceptions and Text Understanding*, (Edited by D. Metzing) pp. 46–61, Walter de Gruyter and Co., Berlin, (1979).

108. S.C. Shapiro and W.J. Rapaport, SNePS considered as a fully intensional propositional semantic network, In *The Knowledge Frontier: Essays in the Representation of Knowledge*, (Edited by Nick Cercone and Gordon McCalla), Springer-Verlag, New York, (1987).

109. L.K. Schubert, Extending the expressive power of semantic networks, *Artificial Intelligence* **7**(2), 163–198, (Summer, 1976).

110. L.K. Schubert, R.G. Goebel and N.J. Cercone, The structure and organization of a semantic net for comprehension and inference, In *Associative Networks: Representation and Use of Knowledge by Computers*, (Edited by N.V. Findler), pp. 121–175, Academic Press, New York, (1979).

111. H.J. Levesque, A formal treatment of incomplete knowledge bases, Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, Toronto, Canada, (1981).

112. H.J. Levesque, The interaction with incomplete knowledge bases: A formal treatment, In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence, (IJCAI-81)*, Vancouver, B.C., Canada, (August, 1981).

113. R.J. Brachman and H.J. Levesque, Expressiveness and tractability in knowledge representation and reasoning, *Computational Intelligence* **3**(2), pp. 78–93, (1987).

114. S.E. Stickel, Automated deduction by theory resolution, In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 1181–1186, Los Angeles, California, August, 1985, An expanded version is available as Technical Note 340, Artificial Intelligence Center, SRI International, (October, 1984).

115. B. Nebel, Computational complexity of terminological reasoning in BACK, *Artificial Intelligence* **34**(3), 371–383, (April, 1988).

116. P.F. Patel-Schneider, H.J. Levesque and R.J. Brachman, ARGON: Knowledge representation meets information retrieval, In *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE Computer Society, Denver, Colorado, (1984).

117. P.F. Patel-Schneider, A decidable first-order logic for knowledge representation, In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 455–458, Los Angeles, CA., (August, 1985). (Also available as AI Tech. Report No. 45, Schlumberger Palo Alto Research, (1985)).

118. D.W. Etherington, On inheritance hierarchies with exceptions, In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pp. 104–108, Philadelphia, PA, (August, 1986).

119. D.A. McAllester, An outlook on truth maintenance, AI Memo 551, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, (August, 1980).

120. D.A. McAllester, Reasoning utility package user's manual, AI Memo 667, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, (April, 1982).

121. R. MacGregor, Loom users manual, Technical Report Working Paper ISI/WP-22, USC/Information Sciences Institute, (1990).

122. J.E. Laird, A. Newell and P.S. Rosenbloom, SOAR: An architecture for general intelligence, *Artificial Intelligence* **33**(1), (September, 1987).

123. C. Peltason, The scheme of Posidonius—Using taxonomic reasoning in design, In *Proceedings Second International Conference on Applications of Artificial Intelligence in Engineering*, pp. 299–314, Cambridge, MA, (August, 1987).

124. E.F. Codd, Data models in database management, In *Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling, ACM SIGMOD* **11**(2), pp. 112–114, (1981).

125. G. Attardi and M. Simi, Consistency and completeness of OMEGA, a logic for knowledge representation, In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pp. 504–510, Vancouver, B.C., Canada, (1981).

126. G. Attardi and M. Simi, A description oriented logic for building knowledge bases, In *Proceedings of the IEEE* **74**(10), (October, 1986).

127. C. Rich and R.C. Waters, Automatic programming: Myths and prospects. *IEEE Computer* **21**(8), 40–51, (August, 1988).

128. C. Rich and R.C. Waters, The programmer's apprentice: A research overview. *IEEE Computer* **21**(11), 11–25, (November, 1988).

129. D. McAllester, R. Givan and T. Fatima, Taxonomic syntax for first order inference, In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR89)*, pp. 289–300, (1989).

130. H. Aït-Kaci, A Lattice theoretic approach to computation based on a calculus of partially ordered type structures, Ph.D. Thesis, Dept. of Computer Science, Univ. of Pennsylvania, Phila., PA, (1984).

131. H. Aït-Kaci and R. Nasr, LOGIN: A logic programming language with built in inheritance, *Journal of Logic Programming* 3(3), 185–215, (1986).

132. G. Smolka and H. Aït-Kaci, Inheritance hierarchies: Semantics and unification, *Journal of Symbolic Computation* 7(3/4), 343–370, (1989).

133. P.F. Patel-Schneider, Undecidability of subsumption in NIKL, *Artificial Intelligence* **39**(2), (June, 1989).

134. M. Schmidt-Schauß, Subsumption in KL-ONE is undecidable, In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR89)*, (Edited by R.J. Brachman, H.J. Levesque and R. Reiter), pp. 421–431, Toronto, Ontario, (May, 1989).

135. G. Smolka, Feature constraint logics for unification grammars, IWBS Report 93, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, (November, 1989), In the *Proceedings of the Workshop on Unification Formalisms—Syntax, Semantics and Implementation, Titisee, (April, 1988)*, MIT Press, (to appear).

136. B. Nebel and G. Smolka, Representation and reasoning with attributive descriptions, In *Sorts and Types in Artificial Intelligence*, (Edited by K.-H. Bläsius, U. Hedtstück and C.-R. Rollinger), Springer-Verlag, Berlin, Germany, (1990); Also available as IWBS Report 81, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, (September, 1989).

137. F. Donini, B. Hollunder, M. Lenzerini, A.M. Spaccamela, D. Nardi and W. Nutt, The frontier of tractability for concept description languages, Dfki-report, DFKI, Postfach 2080, 6750 Kaiserslautern, Germany, (1989).

138. B. Hollunder, Subsumption algorithms for some attributive concept description languages, Masters Thesis, FB Informatik, Universtät Kaiserslautern, 6750 Kaiserslautern, Germany, (1989).

139. G. Smolka, A feature logic with subsorts, LILOG Report 33, IWBS, IDM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, (May, 1988).

140. B. Nebel, Terminological reasoning is inherently intractable, *Artificial Intelligence* **43**(2), (May, 1990), Also available as IWBS Report 82, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, (October, 1989).

141. P.F. Patel-Schneider, A four-valued semantics for terminological logics, *Artificial Intelligence* **38**(3), 319–351, (April, 1989).

142. I.J. Haimowitz, Using NIKL in a large medical knowledge base, Technical Report MIT/LCS/TM-348, Laboratory for Computer Science, Massachusetts Institute of Technology, (January, 1988).

143. S.W. Smoliar and W.R. Swartout, A report from the frontiers of knowledge representation, Draft paper, (October, 1988).

144. J. Doyle and R.S. Patil, Two theses of knowledge representation: language restrictions, taxonomic classifications and the utility of representation services, *Artificial Intelligence* **48**(3), 261–297, (April, 1991).

145. A.R. Haas, A syntactic theory of belief and action, *Artificial Intelligence* **28**(3), 245–292, (May, 1986).

146. J.F. Allen and C.R. Perrault, Analyzing intention in utterances, *Artificial Intelligence* **15**(3), 143–178, (1980), Reprinted in *Readings in Natural Language Processing*, (Edited by B. Grosz, K.S. Jones and B.L. Webber), Morgan-Kaufman, San Mateo, CA, (1986).