*Krypton clearly distinguishes between definitional and factual information by using both frame-based and logic-based languages. The result is a system defined in functional not structural terms.*

# Krypton: A Functional Approach to Knowledge Representation

**Ronald J. Brachman, Fairchild Laboratory for Artificial Intelligence Research**
**Richard E. Fikes, Xerox Palo Alto Research Center**
**Hector J. Levesque, Fairchild Laboratory for Artificial Intelligence Research**

Although much of the current work in knowledge representation is fraught with disagreement, some trends seem to be emerging. In particular, a great deal of effort has focused on developing "frame-based" languages with the following features:

- The principal representational objects, or frames, are nonatomic *descriptions* of some complexity.
- Frames are defined as *specializations* of more general frames.
- Individuals are represented by *instantiations* of generic frames.
- The resulting connections between frames form *taxonomies.*

The widespread appeal of frame taxonomies seems due to how closely they match our intuitions about how to structure the world (as illustrated in folk taxonomies, for example). They also suggest enticing directions for processing (inheritance, defaults, etc.) and have found applications in other areas of computer science, such as database management and object-oriented programming.

While the basic ideas of frame systems are straightforward, complications arise in their design and use. These difficulties typically arise because (1) structures are interpreted in different ways at different times (the principal ambiguity being between definitional and factual interpretations) and (2) the meaning of the representation language is specified only in terms of the data structures used to implement it (typically inheritance networks).

We have developed a design strategy for avoiding these types of problems and have implemented a representation system based on it. The system, called Krypton, clearly distinguishes between definitional and factual information. In particular, Krypton has two representation languages, one for forming descriptive terms and one for making statements about the world using these terms. Further, Krypton provides a *functional* view of a knowledge base, characterized in terms of what it can be asked or told, not in terms of the particular structures it uses to represent knowledge.

## The trouble with frames

As we have noted, a recurring theme in various knowledge-representation languages is the taxonomy of structured descriptions. In such a language, we might have the following description of a family:

```
family
  IS-A social social-structure
  male-parent: man (exactly 1)
  female-parent: woman (exactly 1)
  child: person
```

Even uninterpreted, this type of data structure is undeniably useful. For knowledge representation, however, we must impose an interpretation on the representational objects—that is, they have to *mean* something. As it turns out, we can make an extraordinary number of interpretations of links and nodes in a taxonomy of frames, and typical frame systems leave much of the semantical work to the reader. (The article on p. 30 of this issue gives a compendium of the ambiguous interpretations one finds for the IS-A link.[1])

Yet, despite the multitude of possible interpretations, two approaches to the meaning of frames seem to recur. In the first, frames are *assertions* or statements about the way things are in the world. Under this interpretation, which P. Hayes investigates in some depth,[2] the presence


0018-9162/83/1000-0067$01.00 © 1983 IEEE

of the "family" frame in a frame system would be understood as asserting that every family is a social structure with a male parent, a female parent, and some number of children.

Unfortunately, the assertional point of view turns out to be quite restrictive, principally for two reasons. The first is that instantiation (filling in the slots of a frame), the basic form of assertion in frame systems, makes expressions of incomplete knowledge either difficult or impossible. For example, a statement such as "either Elsie or Bessie is the cow standing in Farmer Jones's field" cannot be made in a typical assertional frame system.

The second reason is that truly composite descriptions cannot be expressed. For example, instead of being able to form a description "A family with no children," we can only create a childless-family frame and assert that families of this type have no children, as if it were an incidental property like having both parents working. (Space does not permit an in-depth treatment of these issues here; the distinction between *assertional* and *structural* links,[3] the inadequacy of instantiation as an assertional mechanism,[4] and the failure of frame systems to handle structured descriptions[1] are discussed in detail elsewhere.)

Frustration with the limitations of viewing frames as assertions might lead us to adopt the other predominant view of frames, as *descriptions* that have no direct assertional import. Representation languages such as KL-One[5] and others take the view that frames and the links between them make up the structure of descriptions; some other mechanism is needed to use the descriptions to state facts. Under this interpretation, the symbol "family" in our example would be taken as an abbreviation for a description such as "a social structure with, among other things,* a male parent who is a man, a female parent who is a woman, and some number of children, all persons." Those who support this interpretation of frames claim that it produces a cleaner language, which does not suffer from the problems plaguing strictly assertional frame systems.

Unfortunately, the cleanliness of the nonassertional approach does not quite allay *all* the fears of misinterpreting links and frames in these systems. What we get with a frame system—even a strictly interpreted structural one—is a package for manipulating symbolic *data structures*. Consequently the user or the user program can draw unwarranted conclusions. Consider, for example, Figure 1. We might think it is easy to answer the question, "How many kinds of rocks are there?" by simply looking at the structure. All we would have to do, apparently, is to count the nodes immediately below "rock." As Figure 2 illustrates, however, counting kinds in a structural network is meaningless, since the language allows the formation of an arbitrary number of descriptive terms such as "large, gray igneous rock" to be on a par with "igneous rock."

A similarly seductive phenomenon arises from the user's access to links in a network. Figure 3 shows two different representations of "bachelor." In the two cases, the distances between "bachelor" and "person" are different. Spreading activation theories of processing in semantic nets might consider this distance to be significant. (The distance between nodes is also considered significant in the inheritance of default properties in which a search is made for the "closest" value.) The links in a nonassertional frame system, however, are simply for forming terms and have no contingent assertional consequence or psychological import.

In sum, then, we seem to be faced with at least two serious problems in our frame-representation systems. We must be careful to interpret structures unambiguously, so that we do not in one breath interpret a link as making an assertion and in the next interpret it as part of the meaning of a term. Even when we are extremely careful about interpreting these structures, however, we must still deal with the fundamental problem of having only data structures to manipulate. We are thus prey to unwarranted inferences in counting data structures or assuming that their presence or absence means something.

*Whether or not the frame expresses sufficiency conditions is essentially independent of the choice to read it structurally or assertionally.

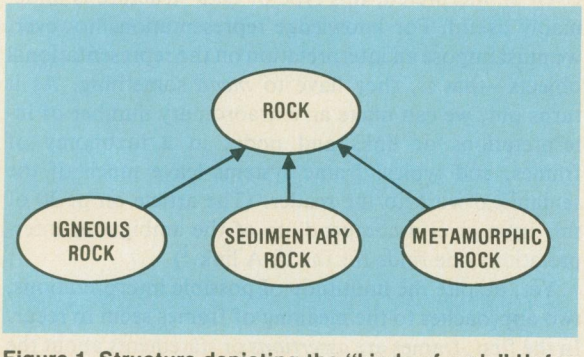



**Figure 1. Structure depicting the "kinds of rock." Unfortunately, as Figure 2 illustrates, we cannot count kinds using a structural network.**

## Krypton: a functional approach

Over the last year, we have been designing and implementing an experimental knowledge-representation system, called Krypton. The primary goal of our effort is to avoid the kinds of problems engendered by the more traditional structure-oriented approaches. We focused on a *functional* specification of the knowledge base, replacing any question like "What structures should the system maintain for a user?" with one that asked "What exactly should the system *do* for a user?" In other words, we decided what kind of *operations* were needed for interacting with a knowledge base without assuming anything about its internal structure. By making only those operations available to a Krypton user, we felt we could control precisely how the system would be used while having the freedom to implement the operations in any convenient way.

Of course, taking a functional view buys us nothing if the operations we provide are precisely the same as the standard ones. In such a case, our system would succumb to the same confusions between structural and assertional uses mentioned earlier. To avoid this problem, we have split the operations into two separate kinds,

yielding two main components for our representation system: a terminological one, or *T Box,* and an assertional one, or *A Box.* The T Box allows us to establish taxonomies of structured terms and answer questions about analytical relationships among these terms; the A Box allows us to build descriptive theories of domains of interest and to answer questions about those domains.

The separation between the two components arises naturally in the two kinds of expressions used to represent knowledge—(nominal) terms and sentences. The T Box deals with the formal equivalent of *noun phrases* such as "a person with at least three children," and understands that this expression is subsumed by (the formal version of) "a person with at least one child," and is disjoint from "a person with at most one child." The A Box, on the other hand, operates with the formal equivalent of *sentences* such as "Every person with at least three children owns a car" and understands the implications (in the logical sense) of assertions such as this one. Furthermore, just as a user has no way to specify after the fact what sentences are logical consequences of others in the A Box, he has no way to specify after the fact where a term fits in a taxonomy in the T Box. The subsumption and disjointness relationships are based only on the structure of the T Box terms and not on any (domain-dependent) facts maintained by the A Box.

In the following sections, we describe the T Box and A Box languages, the operations that are available on these languages, and how these operations are being implemented in Krypton.

**Two languages for representation.** Given its division of representational labor, Krypton can afford to take a strict view of the constructs in its two languages. The expressions in the T Box language are used as structured descriptions and have no direct assertional import. The A Box language, on the other hand, is used strictly for assertions.

*The language of the terminological component.* Rather than simply using an existing frame language in our T Box, we have chosen to break out the primitives

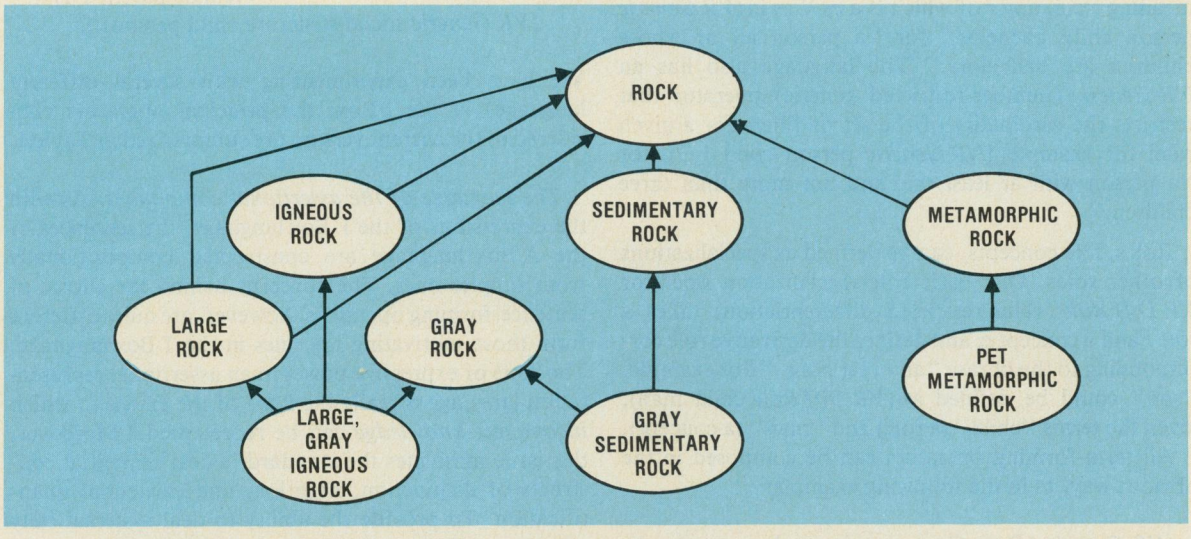



**Figure 2. Figure 1 rock terms rapidly multiply because the representation language allows the formation of descriptive terms like "large rock" and "gray rock."**

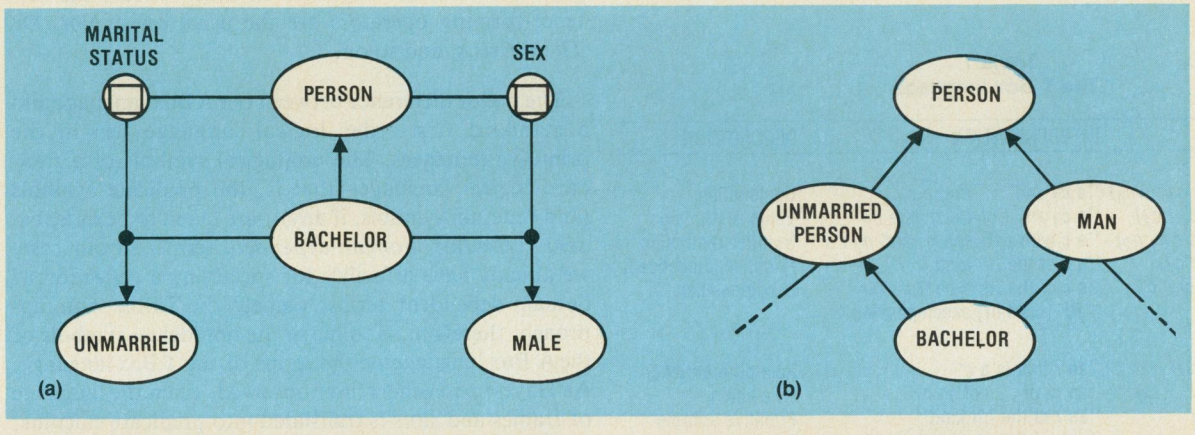



**Figure 3. Two representations of "bachelor." Note that the conceptual distance between bachelor and person in (a) is different from that in (b).**

that seem to make up the essence of frames, to carefully specify their meanings, and to put them together in a compositional framework. In particular, the T Box supports two types of expressions: concept expressions, which correspond roughly to frames (or, more closely, KL-One concepts), and role expressions, the counterparts of slots (or KL-One roles).

In general, concepts and roles are formed by combining or restricting other concepts and roles. For example, the language includes an operator *Conj Generic* (for conjoined generic), which takes any number of concepts, and forms the concept corresponding to their conjunction. This operator could be used to define the symbol "bachelor" by assigning it the expression (*Conj Generic* unmarried-person man), assuming that the symbols unmarried-person and man had appropriate definitions as concepts. (As we will describe later, expressions can be assigned as definitions to atomic symbols. This use of defined symbols is purely for the user's convenience, however.) Concepts can also be formed by restricting other concepts using roles. For example, Krypton has a *VR Generic* (value-restricted generic) operator that takes two concepts $c_1$ and $c_2$ and a role $r$ and yields the term meaning "a $c_1$ any $r$ of which is a $c_2$," as in (*VR Generic* person child bachelor), for "a person all of whose children are bachelors." The language also has an *NR Generic* (number-restricted generic) operator that restricts the cardinality of the set of fillers for a given role; for example, (*NR Generic* person child 1 3), for "a person with at least one and not more than three children."

Roles, like concepts, can be defined as specializations of other roles. One basic role-specialization operator *VR Diff Role* (value-restricted differentiation) takes a role $r$ and a concept $c$, and defines the derivative role corresponding to the phrase "an $r$ that is a $c$." For example, "son" could be defined as (*VR Diff Role* child man), given the terms "child" (a role) and "man" (a concept).

All term-forming operators can be composed in the obvious way, as in the following example:

(*VR Generic* (*Conj Generic* unmarried-person man)
(*VR Diff Role* sibling man)
(*NR Generic* person child 1 ∞)).

This expression can be read as "a bachelor whose brothers have children" or, more literally, "an unmarried person and a man all of whose siblings that are men have between one and ∞ children."

In many domains we want to give necessary but not sufficient conditions for a definition. To this end, Krypton includes facilities for specifying "only-if" definitions. The *Prim Generic* and *Prim Role* operators are used to form primitive specializations of a concept or role. A primitive concept is subsumed by its superconcept, but no sufficient conditions are given for determining if something is described by it.

To see how the T Box language relates to a language of frames, consider the "family" frame used as an example earlier. As a concept, this frame might be expressed in Krypton as

(*Prim Generic* (*Conj Generic*
(*NR Generic*
(*VR Generic* social-structure male-parent man)
male-parent 1 1)
(*NR Generic*
(*VR Generic* social-structure female-parent woman)
female-parent 1 1)
(*VR Generic* social-structure child person))).

We have been experimenting with several different languages in the T Box; the principal operators considered in the current version are summarized in Table 1.

*The language of the assertional component.* As with the expressions of the T Box language, the sentences of the A Box language are constructed compositionally from simpler ones. The concerns behind the choice of sentence-forming operators, however, are quite different from those motivating the ones in the T Box language. The issue of expressive power in an assertional representation language is really the issue of the extent to which *incomplete knowledge* can be represented.[6] Moreover, this issue motivates the standard logical sentential constructs of disjunction, negation, and existential quantification. To provide the ability to deal systematically with incomplete knowledge and to compensate for the fact that the T Box has been purged of any assertional ability, our A Box language is structured like a first-order predicate calculus language. In other words, the sentence-forming operators are the usual ones: *Not, Or, There Exists,* and so on.

The major difference between our A Box language and a standard first-order logical language lies in the primitive sentences. The nonlogical symbols of a standard logical language—that is, the predicate symbols (and function symbols, if any)—are taken to be *independent, primitive, domain-dependent* terms. In our case, we already have a facility for specifying a collection of domain-dependent terms; namely the T Box. Our approach, therefore, is to make the nonlogical symbols of the A Box language be the terms of the T Box language. As Hayes[2] and others have observed, when the language of frames and slots is translated into predicate calculus, the frames and slots become one- and two-place predicates, respectively. The main difference between what

**Table 1.**
**The T Box language.**

| EXPRESSION | INTERPRETATION | DESCRIPTION |
|---|---|---|
| Concepts | | |
| (*Conj Generic* $c_1 \ldots c_n$) | "a $c_1$ and ... and a $c_n$" | Conjunction |
| (*VR Generic* $c_1$ $r$ $c_2$) | "a $c_1$ any $r$ of which is a $c_2$" | Value restriction |
| (*NR Generic* $c$ $r$ $n_1$ $n_2$) | "a $c$ with between $n_1$ and $n_2$ $r$'s" | Number restriction |
| (*Prim Generic* $c$ $i$) | "a $c$ of the $i$th kind" | Primitive subconcept |
| (*Decomp Generic* $c$ $i$ $j$ *disjoint?*) | "a $c$ of the $i$th type from the $j$th [disjoint] decomposition" | Decomposition |
| Roles | | |
| (*VR Diff Role* $c$ $r$) | "an $r$ that is a $c$" | Role differentiation |
| (*Role Chain* $r_1 \ldots r_n$) | "an $r_n$ of ... of an $r_1$" | Role chain |
| (*Prim Role* $r$ $i$) | "an $r$ of the $i$th kind" | Primitive subrole |
| (*Decomp Role* $r$ $i$ $j$ *disjoint?*) | "an $r$ of the $i$th type from the $j$th [disjoint] decomposition" | Decomposition |

these researchers are suggesting and what we have done is that the resulting predicates are not primitive; they can be definitionally related to each other independently of any theory expressed in the A Box language.

Overall, a Krypton system has the structure shown in Figure 4: a T Box of structured terms organized taxonomically, an A Box of (roughly) first-order sentences whose predicates come from the T Box, and a symbol table maintaining the names of the T Box terms so that a user can refer to them.

**Operations on the components.** So far, we have described the Krypton T Box and A Box in terms of two distinct but interconnected languages without saying what a user actually *does* with expressions in these languages. Figure 4 is a somewhat misleading depiction of Krypton's structure, since a user does not have access to either a network in the T Box or a collection of sentences in the A Box. What a user does get access to is a certain fixed set of operations over the T Box and A Box languages. All interactions between a user and a Krypton knowledge base are mediated by these operations.

The operations on a Krypton knowledge base can be divided into two groups: *Tell* operations, which are used to augment a knowledge base and *Ask* operations, which are used to extract information. In either case, the operation can be definitional or assertional.

In terms of the A Box, *Tell* takes an A Box sentence and asserts that it is true. The effect is to change the knowledge base into one whose theory of the world implies that sentence. *Ask* takes a sentence and asks if it is true. The result is determined on the basis of the current theory held by the knowledge base *and* the vocabulary used in the sentence, as defined in the T Box. Schematically, we can describe these operations by

A Box:

*Tell*: KB × Sentence → KB
(Sentence is true.)
*Ask*: KB × Sentence → {yes, no, unknown}
(Is sentence true?)

For the T Box, *Tell* takes a symbol and associates it with a T Box term (noun phrase). The effect is to change the knowledge base into one whose vocabulary includes the symbol defined by the term. We have focused on two *Ask* operations: the first asks whether one T Box term subsumes another, and the second whether one T Box term is conceptually disjoint from another. Schematically, this gives us

T Box:

*Tell*: KB × Symbol × Term → KB
(By symbol, I mean term.)
$Ask_1$: KB × Term × Term → {yes, no}
(Does $term_1$ subsume $term_2$?)
$Ask_2$: KB × Term × Term → {yes, no}
(Is $term_1$ disjoint from $term_2$?)

Of course, there have to be additional *Ask* operations on a knowledge base. So far, for instance, we cannot get anything but a yes/no answer. In the A Box, we have to be able to find out what individuals have a given property; in the T Box, we need some way of getting information from the definitions that is not provided by the subsumption and disjointness questions (e.g., the number of "angles" of a "triangle").

The service provided by Krypton as a knowledge-representation system is completely specified by these operations. In particular, the notions of a taxonomy or a set of first-order clauses in normal form are not part of the interface provided by the system. The actual symbolic structures used by Krypton to realize *Tell* and *Ask* are not available to the user. In this functional view, a knowledge base is treated like an *abstract data type,* characterized by a set of operations rather than by a certain implementation structure.

**Building Krypton.** The following discussion is a generalized accounting of how we are currently constructing Krypton. Because the T Box can be considered to support the A Box, we will discuss the A Box first.

*Making an A Box.* The first thing to notice about an implementation of the A Box is that the expressive power of the assertional language necessitates the use of very general reasoning strategies to answer questions. Specifically, we cannot limit ourselves to the special-purpose methods typical of frame-based representation systems.

For example, to find out if a cow is in the field, we cannot simply search for a representational object standing for it (i.e., an instantiation of the "cow-in-the-field" concept), since, among other things, we may not know all the cows or even how many there are. Yet, we may very well have been told that Jones owns nothing but cows and that at least one of his animals has escaped into the field.

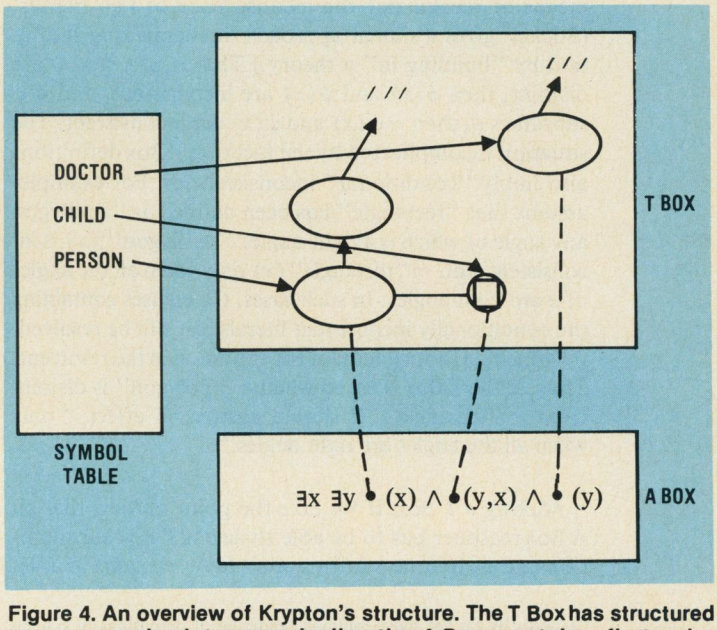



**Figure 4. An overview of Krypton's structure. The T Box has structured terms organized taxonomically, the A Box contains first-order sentences whose predicates come from the T Box, and the symbol table maintains the names of T Box terms.**

The second point worth noticing about the A Box is that if the predicate symbols are indeed T Box terms, then the A Box reasoner needs to have access to the T Box definitions of those terms. For example, once told that Elsie is a cow, the A Box should know that Elsie is an animal and is not a bull. A Box predicates are not simply unconnected primitives (as in first-order logic), so if we want to use standard first-order reasoning techniques, we have to somehow make the connections implied by the T Box.

Conceptually, the simplest way to make these connections is to make the definition of a T Box term assert a sentence in the A Box and then to perform standard first-order reasoning over the resulting expanded theory. For example, after defining the concept "cow," we could automatically assert sentences saying that every cow is an animal and that cows are not bulls, as if these were observed facts about the world. As far as the A Box is concerned, the definition of a term would be no more than the assertion of a "meaning postulate."*

In some sense, this approach would yield a hybrid system like the kind discussed by C. Rich,[7] since we would have two notations stating the same set of facts. Our goal, however, is to develop an A Box reasoner that avoids such redundancies, maintains the distinction between definitional and assertional information, and provides a significant gain in efficiency over simply asserting the meaning postulates as axioms. To this end, we are developing extensions to standard inference rules that take into account dependencies among predicates that can be derived from T Box definitions.

For example, the reasoning of a standard resolution theorem prover depends on noticing that an occurrence of $\phi(x)$ in one clause is inconsistent with $\neg\phi(x)$ in another. Given that realization, the two clauses can be used to infer a resolvent clause. The scope of this inference rule can be increased by using subsumption and disjointness information from the T Box as an additional means of recognizing the inconsistency of two literals. (Stickel[8] gives a similar approach to augmenting resolution by "building in" a theory.) That is, if $\phi$ and $\psi$ are disjoint, then $\phi(x)$ and $\psi(x)$ are inconsistent, and if $\phi$ subsumes $\psi$, then $\neg\phi(x)$ and $\psi(x)$ are inconsistent. The situation is complicated by the fact that T Box definitions also imply "conditional" inconsistencies. For example, assume that "rectangle" has been defined as "a polygon any angle of which is a right-angle." "Polygon" $(x)$ is inconsistent with $\neg$"rectangle" $(x)$ only when all the angles of $x$ are right angles. In such cases, the clauses containing the conditionally inconsistent literals can still be resolved, if we include the negation of the condition in the resolvent. Thus, if the T Box is asked whether "polygon" is disjoint from $\neg$"rectangle," it should answer, in effect, "only when all the angles are right angles."

*Making a T Box.* If we take the point of view that an A Box reasoner has to be able to access T Box subsumption and disjointness information *between* steps in a deduction, we have to be very careful about how long it takes to compute that information. Absolutely nothing will be gained by our implementation strategy if the T Box operations are as hard as theorem proving; we might just as well have taken the standard "meaning postulate" route. We are taking three steps to ensure that the T Box operations can be performed reasonably quickly with respect to the time needed by the A Box.

The first and perhaps most important limit on the T Box operations is provided by the T Box language itself. We can certainly imagine wanting a language that would allow arbitrary "lambda-definable" predicates to be specifed. The trouble is that no complete algorithm for subsumption would then be possible, much less an efficient one. By restricting the T Box language to what might be called the "frame-definable" predicates (in terms of operators such as those already discussed), we stand a chance of getting a usable algorithm and still providing a set of term-forming facilities that have been found useful in AI applications.

The situation is far from resolved, however. The computational complexity of term subsumption seems to be extremely sensitive to the choice of term-forming operators. For example, given a simple T Box language without the *VR Diff Role* operator, the term-subsumption algorithm will apparently be $O(n^2)$ at worst; *with* this operator, however, the problem is not likely to have a nonexponential solution.[9]

As a second step towards fulfilling the efficiency requirement for the T Box, we have adopted a caching scheme in which we store subsumption relationships for symbols defined by the user in a tree-like data structure. We are, in effect, maintaining an explicit taxonomy of the defined symbols. We are also developing methods for extending this cache to include both absolute and conditional disjointness information about T Box terms. The key question, which has not yet been answered, is how to determine a useful subset of all the conditional relationships that could be defined between the symbols.

As final step towards an efficient T Box, we have adopted the notion of a *classifier* much like the one present in KL-One,[10] wherein a background process sequentially determines the subsumption relationship between the new symbol and each symbol for which it is still unknown. Because the taxonomy reflects a partial ordering, we can incrementally move the symbol down toward its correct position. In this way, the symbol taxonomy slowly becomes increasingly informed about the relationship of a symbol to all the other defined symbols.

One important thing to notice about an implementation strategy based on a taxonomy and classification is that it is *only* an implementation strategy. The meaning of the T Box language and the definition of the T Box operators do not depend at all on the taxonomy or on how well the classifier is doing at some point.

**P**roblems arise when frames are used as a representation language: structural and assertional facilities are often confused, the expressive power is limited (particularly when instantiation is the principal oper-

*Indeed, by far the most common rendering of definitions in systems based on first-order logic is as assertions of a certain form (universally quantified biconditionals), a treatment that fails to distinguish them from the more arbitrary facts with the same logical form.

ation), and frame systems are defined only in terms of the data structures used to implement them.

The Krypton system represents an attempt to deal directly with these problems in terms of a strict functional design strategy. By severely limiting the interface between a user and a knowledge base, certain misuses of the system can be minimized. A user is forced to concentrate on what his knowledge base is for, rather than on the implementation details supporting this functionality.

Krypton also advocates that a representation system be divided into two distinct components: terminological and assertional. The terminological component supports the formation of structured descriptions organized taxonomically, while the assertional component allows these descriptions to be used to characterize some domain of interest. In either case, we have a compositional language that is used to interact with a knowledge base.

The Krypton system is now being implemented in Interlisp-D. As of this writing, we have implemented the operations of the terminological component using the taxonomy/classification methodology and are currently investigating its interaction with a version of the Stickel theorem-prover.[11] ■

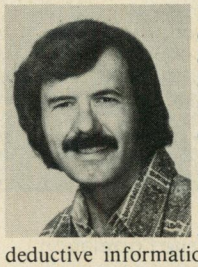
## Acknowledgments

Many people have made significant contributions to this work. Much of Krypton is derived from KL-One, whose development was strongly influenced by Rusty Bobrow, David Israel, Jim Schmolze, and Bill Woods. We also thank Bill Mark, Tom Lipkis, Phil Cohen, and especially Danny Bobrow, Austin Henderson, and Mark Stefik for their participation in many discussions on Krypton and their help in the design of the incipient system. Thanks also to Mark Stickel for the use of his theorem-prover and for the time he spent explaining it to us.

## References

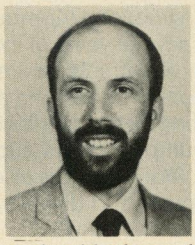
1. R. J. Brachman, "What IS-A Is and Isn't," *Computer,* Vol. 16, No. 10, Oct. 1983, pp. 30-36.
2. P. J. Hayes, "The Logic of Frames," *Frame Conceptions and Text Understanding*, D. Metzing, ed., Walter de Gruyter and Co., Berlin, 1979, pp. 46-61.
3. W. A. Woods, "What's in a Link?: Foundations for Semantic Networks," *Representation and Understanding*, D. G. Bobrow and A. M. Collins, eds., Academic Press, New York, 1975, pp. 35-82.
4. H. J. Levesque, *A Formal Treatment of Incomplete Knowledge Bases,* tech. report 3, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, Calif., Feb. 1982.
5. *Proc. 1981 KL-ONE Workshop,* J. G. Schmolze and R. J. Brachman, eds., tech. report 4, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, Calif., May 1982.
6. R. J. Brachman and H. J. Levesque, "Competence in Knowledge Representation," *Proc. AAAI,* 1982, pp. 189-192.
7. C. Rich, "Knowledge Representation Languages and Predicate Calculus: How to Have Your Cake and Eat It Too," *Proc. AAAI,* 1982, pp. 192-196.
8. M. E. Stickel, "Theory Resolution: Building-In Nonequational Theories," *Proc. AAAI,* 1983.
9. H. J. Levesque, "Some Results on the Complexity of Subsumption in a Frame-based Language," Fairchild Laboratory for Artificial Intelligence, Palo Alto, Calif. (in preparation).
10. J. G. Schmolze and T. A. Lipkis, "Classification in the KL-ONE Knowledge Representation System," *Proc. Int'l Joint Conf. Artificial Intelligence*, 1983.
11. M. E. Stickel, "A Nonclausal Connection-Graph Resolution Theorem-Proving Program," *Proc. AAAI,* 1982, pp. 229-233.

**Ronald J. Brachman** is the author of an article appearing earlier in this issue. His photo and biography are on p. 36.



**Richard E. Fikes** is a member of the Cognitive and Instructional Sciences Group at the Xerox Palo Alto Research Center, where he has been since 1975. Prior to joining PARC, he was a member of the Artificial Intelligence Center at SRI International for six years. Fikes has published numerous papers in automatic planning systems, expert help systems, automation of procedural office work, deductive information retrieval, constraint satisfaction, and knowledge representation, and holds a PhD in computer science from Carnegie-Mellon University. He has served as chairman of Sigart and editor of *Sigart Newsletter* and is on the editorial board of *Decision Support Systems*.

Fikes's address is Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304.



**Hector J. Levesque** is a computer scientist at the Fairchild Laboratory for Artificial Intellience Research. His previous work includes the development of the Procedural Semantic Network formalism, or PSN, and participation in the KL-One project at Bolt Beranek and Newman, Inc., in Cambridge, Mass. Levesque received a PhD from the University of Toronto in 1981. His doctoral dissertation dealt with the problems of incomplete knowledge within a logical representation framework.

Levesque's address is Fairchild Laboratory for Artificial Intelligence Research, 4001 Miranda Ave., Palo Alto, CA 94304.