KRYPTON: Integrating Terminology and Assertion

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

Abstract

The demands placed on a knowledge representation scheme by a knowledge-based system are generally not all met by any of today's candidates. Representation languages based on frames or semantic networks have intuitive appeal for forming descriptions but tend to have severely limited assertional power, and are often fraught with ambiguous readings. Those based on first-order logic are less limited assertionally, but are restricted to primitive, unrelated terms. We have attempted to overcome these limitations in a new, hybrid knowledge representation system, called "KRYPTON". KRYPTON has two representation languages, a frame-based one for forming domain-specific descriptive terms and a logic-based one for making statements about the world. We here summarize the two languages, a functional interface to the system, and an implementation in terms of a taxonomy of frames and its interaction with a first-order theorem prover.

§1 Introduction

Each of the currently predominant styles of knowledge representation seems less than perfect when confronted with the wide range of tasks required of a representation system [2]. For example, frame-like structures are quite popular, but are often imprecisely specified, have limited assertional capabilities (*e.g.*, they tend to be limited to 'instantiation' and defaults), and tend to allow accidental assertions associated with the mere presence or absence of data structures [1]. Logic-based systems are usually more precise and address a much wider range of assertional capabilities, but are limited to primitive, independent predicates; this limitation makes it impossible to syntactically compose new predicates out of old ones.

A productive view would seem to embrace both styles of representation in an attempt to capitalize on the strong points of each. Over the past year, we have been designing and implementing an experimental knowledge representation system—called "KRYPTON"—with exactly this goal in mind.

While the general idea of a hybrid system is not new, KRYPTON is not typical of the predicate calculus/semantic net marriages attempted before. It does not encode logical assertions in network form (as in [4]), nor does it use simultaneous predicate calculus and network representations of the same facts (as in [3] and [12]). Instead, KRYPTON distinguishes between *terminological structure*, expressed in a frame-like taxonomic style, and *assertion*, expressed in a form of predicate calculus. This distinction yields two main components for our representation system: a terminological one (or 'TBox') and an assertional one (or 'ABox'). The TBox allows us to establish taxonomies of structured terms and answer questions about analytical relationships among these terms; the ABox allows us to build descriptive theories of domains of interest and to answer questions about those domains.

Given its division of representational labor, KRYPTON can afford to take a strict view of the constructs in the two languages. The expressions in the TBox language are used as structured descriptions, and have no direct assertional import. Moreover, the ABox language is used strictly for assertions and even universally quantified bi-conditionals have no special definitional import. In what follows, we will describe in more detail the TBox and ABox languages, the operations that are available in KRYPTON and how these operations are being implemented.

§2 Two languages for representation

The two components in KRYPTON reflect the two kinds of expressions it uses to represent knowledge—(nominal) terms and sentences. The TBox contains the formal equivalent of indefinite *noun phrases* such as "a person with at least 3 children", and understands that this expression is subsumed by (the formal version of) "a person with at least 1 child", and is disjoint from "a person with at most 1 child". The subsumption and disjointness relationships among these terms are based only on their structure and not on any (domain-dependent) facts. The ABox, on the other hand, operates with the formal equivalent of *sentences* such as "Every person with at least 3 children owns a car", and understand the implications (in the logical sense) of an assertion such as this one. Here we review the formal constructs that make up the TBox and ABox languages.

2.1 The language of the terminological component

Our TBox language attempts to capture the essence of frames within a compositional and strictly definitional framework, without the ambiguities and possible misinterpretations common in existing frame languages.[1] In particular, the TBox supports two types of expressions: Concept expressions, which correspond roughly to frames, and Role expressions, the counterparts of slots. Thus, the language is defined by a small set of *Concept-* and *Role-forming operators.*

In general, Concepts and Roles are formed by combining or restricting other Concepts and Roles. For example, the language includes an

[1] The TBox is, in large part, a distillation of KL-ONE [13], which accounts for our use below of terms like "value restriction" and "differentiation".

operator **ConjGeneric** ('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[2] *bachelor* by assigning it the expression (**ConjGeneric** *unmarried-person man*) (assuming that the symbols *unmarried-person* and *man* had appropriate definitions as Concepts).

Concepts can also be formed by restricting other Concepts using Roles. For example, KRYPTON has a **VRGeneric** ('value-restricted generic') operator that takes a Concept c_1, a Role r, and a Concept c_2, and yields the term meaning "a c_1 all of whose r's are c_2's", as in (**VRGeneric** *person child bachelor*) for "a person all of whose children are bachelors". The language also has an **NRGeneric** ('number-restricted generic') operator that restricts the cardinality of the set of fillers for a given Role, as in (**NRGeneric** *person child* 1 3) for "a person with at least 1 and not more than 3 children".

Roles, like Concepts, can be defined as specializations of other Roles. One basic Role specialization operator **VRDiffRole** ('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". Thus, this operator would be appropriate for defining *son*, given already defined terms *child* (a Role) and *man* (a Concept), as (**VRDiffRole** *child man*). Another Role-forming operator, **RoleChain**, allows Roles to be functionally composed so that (**RoleChain** *parent brother-in-law*) could be used as the definition of *uncle*.

All of the term-forming operators can be composed in the obvious way, as in, for example,

(**VRGeneric** (**ConjGeneric** *unmarried-person man*)
(**VRDiffRole** *sibling man*)
(**NRGeneric** *person* (**RoleChain** *child child*) 1 ∞)).

This expression can be read as "a bachelor whose brothers have grandchildren" or, more literally, "an unmarried person and a man all of whose siblings that are men are persons whose children have, among them, between 1 and ∞ children".

In many domains one wants to be able to give necessary but not sufficient conditions for a definition. To this end, KRYPTON includes facilities for specifying 'only-if' definitions. The **PrimGeneric** and **PrimRole** operators are used to form primitive specializations of a Concept or Role. A primitive Concept is one that is subsumed by its superConcept, but where no sufficient conditions are given for determining if something is described by it. However, two primitive subConcepts of a Concept are not by definition disjoint nor do they necessarily exhaust that Concept. To introduce a pair of Concepts (or Roles) that do have these properties, the KRYPTON decomposition operators (**DecompGeneric** or **DecompRole**) would be used.

To see how some of the TBox operators relate to a more traditional language of frames, consider the following description of a family (in a hypothetical 'framese'):

family:

isa	*social-structure*
father:	*man* (**exactly** 1)
mother:	*woman* (**exactly** 1)
child:	*person*

Taking this frame as a description of "a social structure with, among other things, a father who is a man, a mother who is a woman, and some number of children, all persons", we might express it in KRYPTON as

(**PrimGeneric** (**ConjGeneric**
(**NRGeneric** (**VRGeneric** *social-structure father man*)
father 1 1)
(**NRGeneric** (**VRGeneric** *social-structure mother woman*)
mother 1 1)
(**VRGeneric** *social-structure child person*))).

We have considered a wide range of operators for the TBox language; the principal ones in the current version are summarized in Table 1.

Expression	Interpretation	Description
Concepts:		
(**ConjGeneric** $c_1 \ldots c_n$)	"a c_1 and ... and a c_n"	conjunction
(**VRGeneric** c_1 r c_2)	"a c_1 any r of which is a c_2"	value restriction
(**NRGeneric** c r n_1 n_2)	"a c with between n_1 and n_2 r's"	number restriction
(**PrimGeneric** c i)	"a c of the i-th kind"	primitive Concept
(**DecompGeneric** c i j *disjoint?*)	"a c of the i-th type from the j-th [disjoint] decomposition"	decomposition
Roles:		
(**VRDiffRole** c r)	"an r that is a c"	differentiation
(**RoleChain** $r_1 \ldots r_n$)	"an r_n of ... of an r_1"	composition
(**PrimRole** r i)	"an r of the i-th kind"	primitive Role
(**DecompRole** r i j *disjoint?*)	"an r of the i-th type from the j-th [disjoint] decomposition"	decomposition

Table 1. The TBox Language.

2.2 The language of the assertional component

As with the expressions of the TBox language, the sentences of the ABox 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 for forming TBox terms. As discussed in [2], the issue of expressive power in an assertional representation language is really the issue of the extent to which *incomplete knowledge* can be represented. Moreover, this issue can be seen to motivate the standard logical sentential constructs of disjunction, negation and existential quantification (see also [9]). So to provide the ability to deal systematically with incomplete knowledge and to compensate for the fact that the TBox has been purged of any assertional ability, our ABox language is structured compositionally like a first order predicate calculus language. In other words, the sentence-forming operators are the usual ones: **Not**, **Or**, **ThereExists**, and so on.

The major difference between our ABox language and a standard first order logical one lies in the atomic sentences. The non-logical symbols of a standard logical language—that is, the predicate symbols (and function symbols, if any)—are taken to be independent, primitive,

[2] As we will describe below, expressions can be assigned as definitions to atomic symbols. This use of defined symbols, however, is purely for the convenience of the user.

domain-dependent terms. In our case, there is already a facility for specifying a collection of domain-dependent terms, namely the TBox. Our approach, therefore, is to make the non-logical symbols of the ABox language be the terms of the TBox language. As observed by Hayes [5] and Nilsson [10], 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 they are suggesting and what we have done is that our predicates are not primitive but are definitionally related to each other (independent of any theory expressed in the ABox language).[3]

§3 Operations on the components

Overall, the structure of KRYPTON can be visualized as in Figure 1: a TBox of roughly KL-ONE-ish terms organized taxonomically, an ABox of roughly first-order sentences whose predicates come from the TBox, and a symbol table maintaining the names of the TBox terms so that a user can refer to them. However, this is a somewhat misleading picture since it suggests that users can manipulate these structures directly. In fact, a user does not get access to either a network in the TBox or to a collection of sentences in the ABox. What a user does get instead is a fixed set of *operations* over the TBox and ABox languages. All interactions between a user and a KRYPTON knowledge base are mediated by these operations.

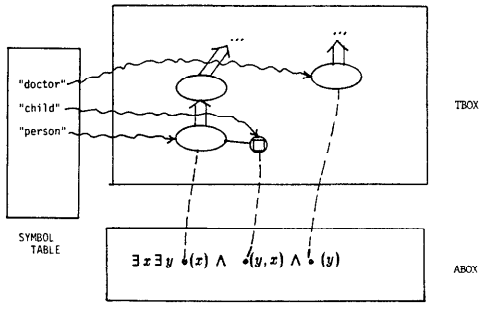

Figure 1. KRYPTON overview.

The operations on a KRYPTON knowledge base can be divided into two groups: the *TELL* operations, used to augment a knowledge base, and the *ASK* operations, used to extract information. In either case, the operation can be definitional or assertional.

In terms of the ABox, the *TELL* operation takes an ABox sentence and asserts that it is true. The effect, roughly speaking, is to change the knowledge base into one whose theory of the world implies that sentence. The corresponding *ASK* operation 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 TBox. Schematically, we can describe these operations by

ABox: *TELL*: KB × SENTENCE → KB
Sentence is true.
ASK: KB × SENTENCE → {*yes*, *no*, *unknown*}
Is sentence *true?*

As for the TBox, the *TELL* operation takes a symbol and associates it with a TBox term (Concept or Role expression). 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 TBox term subsumes another, and the second whether one TBox term is conceptually disjoint from another. Schematically, this gives us

TBox: *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*?*

The TBox *ASK* operations allow a user to inquire about the meaning of the domain-dependent terms being used (without also folding in what is known about the world). In addition, the ABox uses these operations to understand the non-logical terms in the sentences it processes.

Of course there have to be additional *ASK* operations on a knowledge base. For instance, the ones that we have mentioned so far provide no way of getting other than a yes/no answer. In the case of the ABox, we have to be able to find out what individuals have a given property; in the TBox, there has to be some way of getting the information from the definitions that is not provided by the subsumption and disjointness operations (*e.g.*, the fact that the number of *angles* of a *triangle* is necessarily 3).

It is important to stress that the service provided by KRYPTON as a knowledge representation system is completely specified by a definition of the *TELL* and *ASK* 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.[4] The actual symbolic structures used by KRYPTON to realize the *TELL* and *ASK* operations are not available to the user. While it might be useful to think of a knowledge base as structured in a certain way, this structure can only be inferred from the system's behavior. One might consider new operations that allow finer-grained distinctions to be made among knowledge bases,[5] allowing even more of its structure to be deduced, but again it will be the operations that count, not the data structures used to implement them.

One interesting property of this approach to knowledge representation is that there is a difference conceptually between what a system can be *told* (involving expressions in the language used as arguments to *TELL*) and what it has to actually *remember* (involving data structures in the implementation language). This allows us to consider an

[3]The status of function symbols and predicate symbols with more than two arguments is unclear at present. There is no problem incorporating them as *primitives* into the ABox language; the issue is what facilities there should be for relating them definitionally to other terms in the TBox.

[4]In [6], we present a definition of *TELL* and *ASK* that is completely independent of how the knowledge is represented in a knowledge base and is based instead on the set of worlds that are compatible with what is known.

[5]One possibility, for example, is an ABox *ASK* operator that only performs some form of limited inference over what is known.

interface notation that is, in some sense, more expressive than the implementation one, provided that the difference can be accounted for in the translation. In [6], a modal 'auto-epistemic' language is used to supply arguments to *TELL* and yet the resulting knowledge is always represented in first-order terms.

§4 Building KRYPTON

Having discussed the desired functionality of the KRYPTON knowledge representation system, we now sketch in general terms how we are building a system with these capabilitites.

4.1 Making an ABox

The first thing to notice about an implementation of the ABox is that because of the expressive power of the assertional language, very general reasoning strategies will be needed 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 there is a cow in the field, it will not be sufficient to locate 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 Smith owns nothing but cows and that at least one of his animals has escaped into the field.

The second point worth noticing about the ABox is that if the predicate symbols of the ABox language are indeed TBox terms, then the ABox reasoner needs to have access to the TBox definitions of those terms. For example, once told that Elsie is a cow, the ABox should know that Elsie is an animal and is not a bull. However, these facts are not *logical* consequences of the first one since they depend on how the Concept *cow* is defined in the TBox. In general, the issue here is that the ABox predicates are not simply unconnected primitives (as in first-order logic), so that if we want to use standard first-order reasoning techniques, we have to somehow make the connections implied by the TBox.

Conceptually, the simplest way to make the TBox–ABox connection is to cause the act of defining a term in the TBox to assert a sentence in the ABox, and then to perform standard first-order reasoning over the resultant 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 ABox is concerned, the definition of a term would be no more than the assertion of a 'meaning postulate'.[6]

In some sense, this would yield a 'hybrid' system like the kind discussed in [3] and [12], since we would have two notations stating the same set of facts. Our goal, however, is to develop an ABox 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.[7] To this end, we are developing extensions of standard inference rules that take into account dependencies among predicates derivable from TBox 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 expanded by using subsumption and disjointness information from the TBox as an additional means of recognizing the inconsistency of two literals.[8] That is, since *triangle* and *rectangle* are disjoint, *triangle*(x) and *rectangle*(x) are inconsistent; and since *polygon* subsumes *rectangle*, $\neg$*polygon*(x) and *rectangle*(x) are inconsistent. The situation is complicated by the fact that TBox definitions also imply 'conditional' inconsistencies. For example, assume that *rectangle* has been defined as (**VRGeneric** *polygon angle right-angle*). The literal *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 provided that we include the negation of the condition in the resolvent. Thus, if the TBox is asked whether *polygon* is disjoint from $\neg$*rectangle*, it should answer, in effect, "only when all the angles are right angles".

4.2 Making a TBox

Since we take the point of view that an ABox reasoner has to be able to access TBox 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 TBox operations are as hard as theorem proving; we could just as well have gone the meaning postulate route. We are taking three steps to ensure that the TBox operations can be performed reasonably quickly with respect to the ABox.

The first and perhaps most important limit on the TBox operations is provided by our TBox language itself. One can imagine wanting a language that would allow arbitrary 'lambda-definable' predicates to be specified. The trouble is that no complete algorithm for subsumption would then be possible, much less an efficient one. By restricting our TBox language to what might be called the 'frame-definable' predicates (in terms of operators like those we have already discussed—see Table 1), we stand at least a chance of getting a usable algorithm while 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 *very* sensitive to the choice of term-forming operators. For example, it appears that given our TBox language without the **VRDiffRole** operator, the term subsumption algorithm will be $O(n^2)$ at worst; with the **VRDiffRole** operator, however, the problem is as difficult as propositional theorem proving [7].

[6] 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 bi-conditionals), a treatment which fails to distinguish them from the more arbitrary facts that happen to have the same logical form.

[7] There are already precedents in the theorem-proving literature (see [8] and [11]) for using special information about subsumption and disjointness of predicates as a way of guiding a proof procedure.

[8] See [16] for a similar approach to augmenting resolution by 'building-in' a theory.

As a second step towards fulfilling this efficiency requirement for the TBox, 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, in effect, are 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 TBox terms. The key open question regarding these extensions is how to determine a useful subset of the large number of possible conditional relationships that could be defined between the symbols.

As a final step towards an efficient TBox, we have adopted the notion of a *classifier*, much like the one present in KL-ONE [14], 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 towards its correct position. The overall effect of this classification scheme is that the symbol taxonomy slowly becomes more and more informed about the relationship of a symbol to all the other defined symbols.

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

§5 Conclusion

The KRYPTON system represents an attempt to integrate frame-based and logical facilities in such a way as to minimize the disadvantages of each. To do this, KRYPTON separates the representation task into two distinct components: a terminological and an assertional one. The terminological component supports the formation of structured descriptions organized taxonomically; the assertional component allows these descriptions to be used to characterize some domain of interest. The terminological component is a distilled version of frames, characterized by a set of compositional term-forming operators, each with clear import. The assertional component utilizes the power of standard first-order logic for expressing incomplete knowledge. These two components interact through a functionally-specified interface; the user has no access to the structures used to implement it.

An implementation of a KRYPTON system in Interlisp-D is underway. As of this writing, we have implemented the operations of the terminological component using the taxonomy/classification methodology discussed above, and are currently investigating its interaction with a version of the theorem-prover described in [15].

References

[1] Brachman, R. J., Fikes, R. E., and Levesque, H. J., "KRYPTON: A Functional Approach to Representation System," to appear in *IEEE Computer*, September, 1983.

[2] Brachman, R. J., and Levesque, H. J., "Competence in Knowledge Representation," in *Proc. AAAI-82*, Pittsburgh, 1982, 189–192.

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

[4] Fikes, R., and Hendrix, G., "A Network-Based Knowledge Representation and its Natural Deduction System," in *Proc. IJCAI-77*, Cambridge, MA, 1977, 235–246.

[5] Hayes, P. J., "The Logic of Frames," in *Frame Conceptions and Text Understanding*, Metzing, D. (ed.), Walter de Gruyter and Co., Berlin, 1979, 46–61.

[6] Levesque, H. J., "A Formal Treatment of Incomplete Knowledge Bases," Ph. D. thesis, Dept. of Computer Science, University of Toronto, 1981. Also available as FLAIR Technical Report No. 3, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA, February, 1982.

[7] Levesque, H. J., "Some Results on the Complexity of Subsumption in a Frame-based Language," in preparation, 1983.

[8] McSkimin, J. R., and Minker, J., "A Predicate Calculus Based Semantic Network for Deductive Searching," in *Associative Networks: Representation and Use of Knowledge by Computers*, Findler, N. V. (ed.), Academic Press, New York, 1979, 205–238.

[9] Moore, R. C., "The Role of Logic in Knowledge Representation and Commonsense Reasoning," in *Proc. AAAI-82*, Pittsburgh, 1982, 428–433.

[10] Nilsson, N. J., *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA, 1980.

[11] Reiter, R., "Equality and Domain Closure in First-Order Databases," *JACM* **27**, 2 (1980), 235–249.

[12] Rich, C., "Knowledge Representation Languages and Predicate Calculus: How to Have Your Cake and Eat it Too," in *Proc. AAAI-82*, Pittsburgh, 1982, 193–196.

[13] Schmolze, J. G., and Brachman, R. J., *eds.*, "Proceedings of the Second KL-ONE Workshop," FLAIR Technical Report No. 4, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA, May, 1982.

[14] Schmolze, J. G., and Lipkis, T. A., "Classification in the KL-ONE Knowledge Representation System," in *Proc. IJCAI-83*, Karlsruhe, W. Germany, 1983.

[15] Stickel, M. E., "A Nonclausal Connection-Graph Resolution Theorem-Proving Program," in *Proc. AAAI-82*, Pittsburgh, 1982, 229–233.

[16] Stickel, M. E., "Theory Resolution: Building-In Nonequational Theories," in *Proc. AAAI-83*, Washington. D. C., 1983.