AN ESSENTIAL HYBRID REASONING SYSTEM: KNOWLEDGE AND SYMBOL LEVEL ACCOUNTS OF KRYPTON

Ronald J. Brachman
AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974

Victoria Pigman Gilbert
Schlumberger Palo Alto Research
3340 Hillview Ave.
Palo Alto, CA 94304

Hector J. Levesque[1]
Dept. of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 1A7

ABSTRACT

Hybrid inference systems are an important way to address the fact that intelligent systems have muiltifaceted representational and reasoning competence. KRYPTON is an experimental prototype that competently handles both terminological and assertional knowledge; these two kinds of information are tightly linked by having sentences in an assertional component be formed using structured complex predicates denned in a complementary terminological component. KRYPTON is unique in that it combines in a completely integrated fashion a frame-based description language and a first-order resolution theorem-prover. We give here both a formal Knowledge Level view of the user interface to KRYPTON and the technical Symbol Level details of the integration *of* the two disparate components, thus providing an essential picture of the abstract function that KRYPTON computes and the implementation technology needed to make it work. We also illustrate the kind of complex question the system can answer.

I Introduction

Many of today's knowledge representation (KR) systems offer their users a choice of more than one language for expression of domain knowledge. While the idea has been important to the field for many years *(e.g.,* see [Brown and Burton, 1975] [Moses, 1971] [Sloman, 1971]), "multiple representations" seems to have recently become a popular catch phrase. Many of the modern expert system development environments wave the polyglot banner, and except perhaps for some stalwart first-order logicians, most everyone would probably agree that one uniform language will not serve all representational needs.

It is sometimes difficult to discern the true value of multiple languages; some of the commercial development tools seem simply to appeal to "the more the merrier," without any clear idea of how merrier is better. However, on the research front, there have fortunately been some coherent views expressed on the merits of bringing disparate dialects together. The arguments have mainly to do with the naturalness of expressing certain kinds of facts in certain forms (see, *e.g.,* [Rich, 1982]), or with the efficiency of computing some inferences once some datum is massaged into a certain representational form [Genesereth, 1981].[2]

The arguments in favor of naturalness and computational superiority are important ones. However, as we have argued elsewhere [Brachman and Levesque, 1982], there may be a simpler, more basic reason to diversify: an intelligent system has more than one kind of representational need. For example, it is interesting to observe that so far, most KR systems only allow the expression of what we would call *assertional* knowledge—statements of facts or beliefs. If the system uses logic or production rules, the sentential nature of its content is self-evident *(e.g.,* "if the infection is primary bacteremia and ..., then there is suggestive evidence that the organism is bacteroides"). Even if the system uses frames or semantic nets, it is most likely encoding sentences of the form, "elephants are gray", or "Clyde is an armadillo" (see [Brachman, 1985] for more on this).

But what accounts for the meanings of the terms used in these sentences? In addition to representing the beliefs (or, popularly, "knowledge'') of a cognitive system, we need to account at least for the complex terms (like *bacteremia* and *armadillo,* above) used in forming those beliefs. In other words, unless we intend to give up the lexical ghost and take all terms as inscrutable atomic primitives, some substantive representational explanation must be given for the *noun phrases* out of which the sentences expressing beliefs are constructed.

We have designed and implemented a complete hybrid reasoning system, called KRYPTON, that is significantly different from other "multiple representation" systems[3] in that it tries to provide exactly such an explanation. Instead of concentrating on multiple ways to perform the same inference, KRYPTON tries to span two fundamentally different types of representation and reasoning. Not only does it provide a language for composing sentences expressing beliefs (using sentential operators of the usual sort, for conjunction, quantification, negation, etc.), it supports the definition of complex predicates to be used in those sentences (with a set of special term-forming operators, for conceptual conjunction, role composition, role value restriction, etc.). Further, KRYPTON supports an appropriate set of inferences over sentences and over complex terms. Thus, KRYPTON is at least minimally competent in both the *assertional* and *terminological* domains.

In this paper, we will describe KRYPTON in some detail mainly by addressing two questions (both of which are crucial for designers of *any* representation system to answer):

1. *Exactly what service is being provided to the user?* The user should know, at some level not dependent on implementation details, what questions the system is capable of answering, and what operations are permitted that allow new information to be provided to it. He needs to know how questions put to a knowledge base will be answered strictly as a function of the information it contains, and not dependent on how the information is represented. In other words, the user requires what Newell [Newell, 1981] calls a *Knowledge Level* account (see also [Brachman and Levesque, 1984b] [Brachman, *et al.,* 1983b] [Levesque, 1984c]).

2. *How does the implementation realize the service promised?* In other words, how, at the *Symbol Level,* is the Knowledge Level account realised? In the case of a hybrid system, the interface between the disparate parts of the hybrid is particularly important.

[1] Fellow of the Canadian Institute for Advanced Research

[2] For an interesting recent paper on this general subject, see [Sloman, 1984).

'Except for KL-TWO [Vilain, 1985), whose history is significantly intertwined with that of KRYPTON.

This paper provides detailed Knowledge and Symbol Level accounts of KRYPTON (previous accounts [Brachman, *et* a/., 1983a] [Brachman, *et al.,* 1983b] gave only sketchy and informal details of what a real KRYPTON might be like). We first provide a formal semantics that brings together meanings of terms and the *subsumption* relation with meanings of sentences and *truth valuations.* This leads to a Knowledge Level account of KRYPTON in terms of a set *of operations on knowledge bases.* Subsequently, we explain how, at the Symbol Level, our implementation achieves this functionality in terms of an alteration to the meaning of unification in a resolution theorem-prover. In this section, we concentrate on the assertional component and how term definitions affect it, since details of our terminological component have been provided elsewhere [Brachman, *et al*., 1983a]. We conclude the paper with some sample inferences drawn by KRYPTON, to illustrate how the two components are indeed tightly integrated, and that the system provides the kind of hybrid reasoning service promised by our Knowledge Level description.

Before we begin, it should be noted that KRYPTON's answers to the two questions posed above are more complicated and also more interesting than they would be for less ambitious systems. In the case of a "non-essential" hybrid that uses multiple representations of the same facts, Question 1 can be finessed with a simple explication of the semantics of the language into which all of the multiple representations can be translated (hopefully a standard one, as with first-order logic in MRS [Genesereth, 1981]), and an account of the translation rules used among the languages. In such systems, the multiple representations are usually not integrated in any important way, so Question 2 has an easy answer.

However, for an essential hybrid like KRYPTON, which attempts to integrate two fundamentally different kinds of representation and reasoning facilities, the picture is more complex. At the Knowledge Level, KRYPTON provides extra functionality over systems based on standard first-order logic; thus, we cannot ask the user to rely on intuitive knowledge of standard formal semantics, but rather must provide a detailed explanation of how the meanings of sentences in the assertional component can be affected by the structure of descriptions in the terminological component. At the Symbol Level, the user needs to know that, in the implementation he is using, complex term definitions really will have the impact on the meaning of the believed sentences advertised in the Knowledge Level account. It is not enough to say that KRYPTON has a frame-style description language for forming terms and a first-order predicate language for forming sentences—we must explain how the interpretations of the sentences by the theorem-prover depend on the definitions of the terms. As the reader will soon see, both the Knowledge Level and Symbol Level accounts must take seriously KRYPTON's hybrid structure.

II Representational Roots

As hinted above, KRYPTON is a hybrid system with two main components, one that specialises in assertional reasoning (the *A Box),* the other in terminological reasoning (the *TBox).* Each component has its own language, and its own inference mechanism.

KRYPTON developed mainly out of work on KL-ONE, a fairly complex representation system based on both semantic networks and frames [Brachman and Schmolse, 1985]. As KL-ONE evolved, it became evident that its strength lay in its ability to form complex descriptions. To the extent that you could *say* anything at all with KL-ONE, it was with clearly impoverished assertional facilities. In fact, if frame-based systems in general have any advantage, it comes from their ability to form descriptions, rather than sentences. As pointed out in [Brachman and Levesque, 1982] and [Brachman, *et* al., 1983b], logical languages like that of first-order predicate calculus are far superior at meeting the needs for belief representations, since they can provide for a kind of noncommittal expression not possible with frame languages.

Given the strength of KL-ONE's terminological facilities and the weakness of its assertional ones, it seemed best to abandon the latter completely, and adopt a language better suited to sentence representation. While we would no doubt have used a more computationally tractable inference framework than full first-order logic if an appropriate one were available,4 we chose to build our initial KRYPTON ABox on Stickel's connection graph theorem-prover [Stickel, 1982]. While the full first-order resolution mechanism is, in a sense, too powerful for our needs [Levesque, 1984a] [Levesque, 1984b], as a first ABox it turned out to be a remarkably good fit (as will be detailed in Section IV).

The heart of KRYPTON is the connection between the two components: predicates used in the ABox are actually defined in the TBox.6 Thus, all of the analytic inferences computed by the frame-based TBox must be available for consumption in the logic-based ABox. This paper illustrates at both the Knowledge and Symbol Levels how the ABox benefits from the special-purpose definitional facilities available in the TBox.

Finally, as the reader of earlier papers on KRYPTON will note, there are potentially several versions of the TBox language that could be discussed here. While our implemented TBox includes virtually all of the operators mentioned in [Brachman, *et* a/., 1983b],6 not all of those operators are tightly integrated with the ABox. Therefore, we have limited our discussion in this paper to those parts of the TBox that currently can affect the meaning of sentences in the ABox. For example, while subsumption for concepts with number restrictions works in the TBox, it is not treated here, since it awaits full integration with the theorem-prover.7

III Knowledge Level Operations

Although our basic ABox is a standard first-order predicate logic, its proper integration in KRYPTON demands that complex terms defined in the TBox be available for use as predicates in assertions. Thus, if the resulting assertional capability is considered as a predicate logic, then it is a non-standard one, in that it has both the normal sentential operators and special operators used for constructing complex predicates. In this section we will demonstrate the hybrid semantics necessary to explain the integration. First we present the syntax of KRYPTON's languages and their semantics; we then rigorously specify the interface that a Krypton system presents to an outside user. In particular, we

4 We are currently working on a relevance-style (Anderson and Belnap, 1975] limited inference mechanism [Patel-Sehneider, 1086]; see also [Levesque, 1984b] for properties appropriate to a belief representation.

5The mapping between symbols and their definitions is maintained by a *symbol table,* shared by the TBox and ABox. Please consult [Brachman, *et al.,* 1983b] and (Brachman, *et al.,* 1983a] for more on the structure of the system.

*More precisely, we have implemented the language exactly as described In [Brachman, *et al.,* 1983b), except for the *VRDiff* operator, which was ammitted for computational reasons as discussed in |Brachxuan and Levesque, 1984a], and the *DecompRole* operator.

7Also, a note about KL-TWO seems in order here. The terminological language in that system is much more extensive than the one discussed here (it is almost a superset of our TBox). Given that, its computational properties are lets understood, and the completeness of its subsumption mechanism is somewhat in doubt (it is not clear exactly what is currently implemented and working). On the other hand, KL-TWO uses a much more tractable (propositional) ABox. Also, as reported in [Vilain, 1985), the method of integrating terminology and assertion is quite different in KL-TWO.

will precisely define operations which allow questions to be answered about the world or about the conceptual vocabulary being used, as well as operations which allow new information about the world to be accepted and new terms to be defined.

A. Syntax and Semantics

The language currently implemented in the TBox has two main categories: *concepts* and *roles,* roughly comparable to frames and slots. These are inter-defined by the following simple BNF grammar:

$$\begin{aligned}\langle concept\rangle ::= & \ \langle 1\text{-}predicate\text{-}symbol\rangle \\ & \mid (\textbf{ConGeneric}\ \langle concept\rangle_1 \ldots \langle concept\rangle_n),\ n \geq 0 \\ & \mid (\textbf{VRGeneric}\ \langle concept\rangle\ \langle role\rangle\ \langle concept\rangle) \\ \langle role\rangle ::= & \ \langle 2\text{-}predicate\text{-}symbol\rangle \\ & \mid (\textbf{RoleChain}\ \langle role\rangle_1 \ldots \langle role\rangle_n),\ n \geq 1.\end{aligned}$$

This language has much in common with many typical frame languages. *(1 predicate-symbol)* and *{2-predicate-symbol)* are primitive (undefined) concepts and roles. ConGeneric allows the conjunction of concepts. VRGeneric allows the specification of a type restriction on the filler of a role *(e.g.,* (VRGeneric *Paper Author Scientist)* would represent the concept of a paper all of whose authors were scientists). Role Chain supports the composition of two-place relations.

For the ABox, the language we will use is that of a pure (that is, function-free) predicate calculus.[8] The grammar, then, is the following:

$$\begin{aligned}\langle wff\rangle ::= & \ (\langle k\text{-}predicate\text{-}symbol\rangle\ \langle var\rangle_1 \ldots \langle var\rangle_k),\ k \geq 0 \\ & \mid (\textbf{NOT}\ \langle wff\rangle) \\ & \mid (\textbf{OR}\ \langle wff\rangle\ \langle wff\rangle) \\ & \mid (\textbf{EXISTS}\ \langle var\rangle\ \langle wff\rangle).\end{aligned}$$

It is assumed that the other usual logical connectives (conjunction, universal quantification, etc.) can be defined syntactically in terms of the ones provided here (we will use standard logical typography for these when convenient). Note that one- and two-place predicate symbols are both terms of the TBox language and components of the ABox language. To make this intersection explicit, we also define the following categories:

$$\begin{aligned}\langle TBox\text{-}symbol\rangle ::= & \ \langle 1\text{-}predicate\text{-}symbol\rangle \\ & \mid \langle 2\text{-}predicate\text{-}symbol\rangle \\ \langle gsymbol\rangle ::= & \ \langle k\text{-}predicate\text{-}symbol\rangle,\ k \geq 0 \\ \langle gterm\rangle ::= & \ \langle gsymbol\rangle \mid \langle concept\rangle \mid \langle role\rangle\end{aligned}$$

So gterms, as they will be understood here, are either predicate symbols or composite TBox expressions and each gterm has an associated arity (1 for concepts, 2 for roles, and it for each k-place predicate symbol).

The semantics of the TBox and ABox languages is defined in terms of mappings from gsymbols to relations of the same arity over some domain. Given a domain of individuals and such a mapping, it will be possible to specify the extension of every gterm and the truth value of every sentence.[9] For the TBox language, this is defined as follows:

Definition 1 *Let $\mathcal{D}$ be any set. Let $\mathcal{E}$ be any function from gsymbols to relations over $\mathcal{D}$ such that $\mathcal{E}(s)$ has the same arity as s. Then for any gterm e, we define the EXTENSION of e wrt $\mathcal{E}$ by*

1. *The extension of any gsymbol s is $\mathcal{E}(s)$.*
2. *The extension of (**ConGeneric** $e_1 \ldots e_k$) is the intersection of the extensions of the e_i, and $\mathcal{D}$ if k is 0.*
3. *The extension of (**VRGeneric** e_1 r e_2) is those elements x of the extension of e_1 such that $\langle x, y\rangle$ is in the extension of r only when y is in the extension of e_2.*
4. *The extension of (**RoleChain** $r_1 \ldots r_k$) is the relational composition of the extensions of $r_1 \ldots r_k$.*

For example, the extension of (**VRGeneric** *Person Child Doctor*) would be the elements x of the extension of *Person* such that any y such that $\langle x, y\rangle$ is in the extension of *Child* is also in the extension of *Doctor*; that is, the complex term stands for those persons whose children are all doctors.[10] Similarly, the extension of (**RoleChain** *Child Child*) is the set of all pairs $\langle x, z\rangle$ such that for some y, $\langle x, y\rangle$ is in the extension of *Child* and $\langle y, z\rangle$ is also in the extension of *Child*; that is, the expression stands for the *Grandchild* relation.

To define the semantics of the ABox language, we need the notion of an *environment,* which is a function from variables to elements of some domain. Given an environment $\mathcal{V}$, a variable x, and an object o, we also define $\mathcal{V}[x/o]$ to be the environment that is exactly like $\mathcal{V}$ except that x is mapped to o. The truth of a wff is defined in terms of a mapping $\mathcal{E}$ and an environment $\mathcal{V}$ as follows:

Definition 2 *Let $\mathcal{D}$ be any set. Let $\mathcal{E}$ be any function from gsymbols to relations over $\mathcal{D}$ such that $\mathcal{E}(s)$ has the same arity as s. Let $\mathcal{V}$ be any environment over $\mathcal{D}$. Then for any wff α, we define the TRUTH of α wrt to $\mathcal{E}$ and $\mathcal{V}$ by*

1. *$(p\ x_1 \ldots x_k)$ is true iff $\langle \mathcal{V}(x_1), \ldots, \mathcal{V}(x_k)\rangle$ is in the relation $\mathcal{E}(p)$.*
2. *(**NOT** α) is true iff α is not true.*
3. *(**OR** α β) is true iff either α or β is true.*
4. *(**EXISTS** x α) is true iff for some d in $\mathcal{D}$, α is true wrt to $\mathcal{E}$ and $\mathcal{V}[x/d]$.*

One thing to notice about this definition is that the truth of sentences does not depend on an environment at all and so, as with gterms, the meaning of sentences is strictly a function of the set $\mathcal{D}$ and the mapping $\mathcal{E}$.

So far, the only relation between the semantics of the TBox and the ABox languages is that both depend on the assignment $\mathcal{E}$ of relations to gsymbols. However, the coupling is closer than this since TBox symbols can be *defined*, that is, associated with other gterms which become their definitions. The net effect is to constrain the mapping $\mathcal{E}$ so that the extension of the defined symbol is the same as the extension of the gterm. To make this precise, we introduce the notion of a *symbol table* as follows: a symbol table S is a function from 1-place predicates to concepts and 2-place predicates to roles; for any TBox symbol g, $S(g)$ is the gterm which is the definition of g under S. Strictly speaking, we should be careful to avoid circular definitions[11] in a symbol table, but will not do so here. In fact, we will use the convention that a gsymbol is undefined whenever $S(p)$ equals p itself. In other words, for any gsymbol g, we say that g is *primitive* wrt S when g equals $S(g)$.[12]

A key notion given a symbol table S is that of a mapping $\mathcal{E}$ being an *extension function*:

*The actual implementation described below uses function symbols of every arity including constants (O-ary ones). We are omitting these here for simplicity.

* By a *sentence,* we mean a doted wff of the ABox language.

[10] As a matter of terminology, we will say that the fillers of the *Child* role are constrained to be in the extension of *Doctor*.

[11] An example of a circular definition might be one where $S(p)$ is (**ConGeneric** q r) and $S(q)$ is (**ConGeneric** p s).

[12] In particular, we assume that all k-place predicates for k greater than 2 are primitive.

Definition 3 *Let S be a symbol table. $\mathcal{E}$ is an EXTENSION FUNCTION wrt S iff for every gsymbol g, $\mathcal{E}(g)$ is the same as the extension of $S(g)$ wrt $\mathcal{E}$.*

In other words, an extension function is a mapping that obeys the definitions specified by S. For example, if p is defined by S to be the gterm (**ConGeneric** q r), then $\mathcal{E}(p)$, to be an extension function, would have to be the intersection of $\mathcal{E}(q)$ and $\mathcal{E}(r)$.

B. Outcomes and Knowledge Bases

Using the notion of an extension function defined above, we can define what it means to be a *truth valuation* and a *subsumption relationship*, which together will tell us what the world is like and how the gterms relate to each other. We define the first as follows:

Definition 4 *Let S be a symbol table. Let w be a mapping from sentences to $\{true,false\}$. w is a TRUTH VALUATION wrt to S iff there is a set $\mathcal{D}$ and an extension function $\mathcal{E}$ wrt S over $\mathcal{D}$ such that $w(\alpha)$ = true iff α is true wrt $\mathcal{E}$.*

So a truth valuation is a mapping from sentences to truth values that follows the definition of truth given earlier and respects the definitions given by S. For example, if p is defined by S to be q, then any truth valuation that says that (**EXISTS** x (p x)) is true, must also say that (**EXISTS** x (q x)) is also true.

For terms, we define the subsumption relationship as follows:

Definition 5 *Let S be a symbol table. Let e_1 and e_2 be gterms. e_1 SUBSUMES e_2 wrt S iff for any set $\mathcal{D}$ and any $\mathcal{E}$ that is an extension function wrt S, the extension of e_1 is a superset of that of e_2.*

So, for example, the gterm p always subsumes (**ConGeneric** p r), and if q is defined by S to be (**ConGeneric** p r), then p subsumes q with respect to S (for any p, q, and r).

The notion of an *outcome* is that of a complete specification of what the world is like and how the gterms are interrelated. It is defined by the following:

Definition 6 *An OUTCOME is a pair $(\Rightarrow,w)$ where, for some symbol table S,*

1. *$\Rightarrow$ is the subsumption relationship wrt S (so $\Rightarrow \subseteq$ gterm $\times$ gterm)*
2. *w is a truth valuation wrt S (so w is a function from sentences to truth values).*

Note that an outcome does not determine a unique symbol table. As a very simple example, if the relation $\Rightarrow$ says that the gterms p and (**ConGeneric** q r) subsume each other, it could have been the case that p was defined as (**ConGeneric** q r) or as (**ConGeneric** r q) or even as some other expression that subsumes and is subsumed by (**ConGeneric** r q). The actual syntactic form of the definition of p is not considered to be relevant; what counts is the relationship between p and all other gterms.

While an outcome is a complete specification of what the world and vocabulary are like, a *knowledge base* is considered a partial specification of the same. Formally, we define a knowledge base to be any set of outcomes. (This is the usual convention of treating a partial x as the set of all complete x's that are 'consistent' with it.) So the outcomes that are members of a KB are those that are consistent with the information available to the KB; the outcomes that are not members of the KB are those that can be ruled out based on the information available to the KB. So for example, if a KB knows only that α is true and that p is defined to be (**ConGeneric** q r), the outcomes it will contain are all those where p subsumes (**ConGeneric** q r) and vice-versa, and where α is true. This is not to say that the KB does not know about other truths or subsumption relationships. For example, in all the outcomes in the KB, p also subsumes (**ConGeneric** r s q), and (**OR** α β) is also true. Moreover, for any KB at all, all of its outcomes have the property that p subsumes (**ConGeneric** p q), and every valid sentence of logic comes out true.

C. The Operations on Knowledge Bases

We are now in a position to define what operations are available on these abstract knowledge bases. These are the operations that actually define KRYPTON and the only ones that an implementation has to provide to a user.

First, we have a simple operation to get started. NEWKB creates a new knowledge base that includes no definitions or contingent facts. NEWKB is defined this way:

$$\text{NEWKB}[\,] = \{(\Rightarrow,w) \mid (\Rightarrow,w) \text{ is an outcome}\}.$$

In other words, the result of NEWKB is the set of all outcomes. Note that this KB does have some nontrivial properties, such as those mentioned at the end of the last section.

There are two ABox operations provided by KRYPTON. The first, called ASK, is used to determine what the world is like according to what is known. Informally, ASK takes a sentence and a KB and returns 'yes' or 'no'. The second, called TELL, is used to inform the KB of what the world is like. It takes a sentence and a KB and returns a new KB that knows that the sentence is true. More formally,

$$\begin{aligned}&\text{ASK}[\alpha,\text{KB}] = \\ &\quad \text{yes, if for each } (\Rightarrow,w) \text{ in KB, } w(\alpha) = \text{true},\\ &\quad \text{no, otherwise.}\\ &\text{TELL}[\alpha,\text{KB}] = \{(\Rightarrow,w) \text{ in KB} \mid w(\alpha) = \text{true}\}.\end{aligned}$$

So TELL rules out the possibility that its argument is false by retaining only those outcomes where it comes out true. Similarly, ASK answers 'yes' precisely when the possibility that α is false has been ruled out.

The TBox equivalent to TELL is called DEFINE. It takes a TBox symbol, a TBox expression and a KB and tells the KB that the symbol is defined by the expression. The equivalent to ASK is called SUBSUMES. It takes two TBox terms and returns 'yes' or 'no' according to whether the first subsumes the second based on what is known about the terms. These are defined as follows:

$$\begin{aligned}&\text{DEFINE}[g,e,\text{KB}] = \{(\Rightarrow,w) \text{ in KB} \mid g \Rightarrow e \text{ and } e \Rightarrow g\}.\\ &\text{SUBSUMES}[e_1,e_2,\text{KB}] = \\ &\quad \text{yes, if for each } (\Rightarrow,w) \text{ in KB } e_2 \Rightarrow e_1,\\ &\quad \text{no, otherwise.}\end{aligned}$$

The TBox also has operations that have no (current) analogue in the ABox. These operations are questions that return sets of gsymbols, instead of 'yes' or 'no'.[13] The intent is that we should be able to find out what symbols have been defined and to reconstruct, for each such symbol, a definition for it. It may not be the actual definition that was used, but its effect overall would be the same. The three operations are VOCAB, which returns the gsymbols that have been defined; PRIMS, which given a TBox term returns the ultimate primitives[14] that make up the expression; and, ROLEPAIRS, which given a concept, returns a

[13] Stickel's theorem-prover provides answers to some assertional "wh-questions" but the question of an appropriate set for KR purposes is still an open one.

[14] Recall that a primitive is a symbol that has no definition. Thus what constitutes a primitive may vary as definitions are acquired.

list of pairs (c, *p)* where p is a primitive concept and c is a list of primitive roles, and where the gterm constrains the fillers of the chain of *c* to be in the extension of *p.* These operations are defined in the following way:

VOCAB[KB] = {p, a gsymbol | there is a different gsymbol q such that SUBSUMES[q, p, KB] = yes}.
PRIMS[e, KB] =
if e is a concept then {p, a 1-predicate-symbol | p is not in VOCAB[KB] and SUBSUMES[p, e, KB] = yes}
if e is a role then (q_1 ... q_k) where no q_i is in VOCAB[KB] and SUBSUMES[(RoleChain q_1 ... q_k), e, KB] = yes.
ROLEPAIRS[e, KB] = {((q_1, ..., q_k), p) | neither p nor any q_i is in VOCAB[KB] and SUBSUMES[(VRGeneric e (RoleChain q_1 ... q_k) p), e, KB] = yes}.

Note that a gsymbol is a primitive exactly when it is not defined, that is, when it is not in VOCAB.

Given the details of this Knowledge Level account, there are a number of important properties of KRYPTON that we can prove. For example, it is easy to show that ASK and TELL have the right relationship, *i.e.,* that ASKja, TELL(a, KB)] = true, for any KB, as well as any subsequent KB (TELL works *monotonically*—anything that is believed continues to be believed as new facts or new definitions are added). Similarly, it can be shown that DEFINE and SUBSUMES have a proper relationship, i.e.,

SUBSUMES[*, e, DEFINE(s, s, KB)] = yes
and
SUBSUMES[e, s, DEFINE(s, e, KB)] = yes (for any KB).

We can show that all of the logical truths hold in an empty KB, *e.g.,* ASK[p V -p, KB_0] = yes, where KB_0 is the value of NEWKB[ ]. We can show that all of the appropriate subsumption relationships [Brachman and Levesque, 1984a] hold in the TBox, as in, *e.g.,*

SUBSUMES [p_1, (ConGenerIc p_l $p_,$), KB] (for any KB).

It can also be shown that the expanded form of a definition into only primitives (using the results of PRIMS and ROLEPAIRS) is equivalent to the original definition. Finally, and most cogent to the focus of this paper, it can be shown that the ABox and the TBox have the desired relationship. For example, universal assertions that should follow directly from definitions can be shown to do so:

ASK[$\forall x$ *Man*(x) $\supset$ *Person*(x), DEFINE[*Man*, (VRGeneric *Person Sex Male*), KB]] = yes (for any KB).

In genera], because we can prove such things about KRYPTON, this kind of Knowledge Level account plays a vital part in providing a predictable, reliable interface for consumers of the KR service.

IV The KRYPTON Implementation

Having defined the existing KRYPTON interface, we now turn our attention to the way the service is implemented. In general, it is possible to build a hybrid KR system wherein term definitions are expressed in a special language, and then simply translated into sentences in another. This loose kind of integration at least allows a system to take some of the implications of definitions into account. However, integration of this type has two potentially serious problems: first, contingent assertional sentences (even universally quantified biconditional!) simply do not have the same import as definitions and loose integration forces their conflation (contrast "every animal is either not in my field or is a cow" with "every triangle is a three-sided polygon"); second, adding more sentences to a belief-reasoner (i.e., a theorem-prover) will invariably slow it down (see [Stickel, 1985]).

In KRYPTON, we have embodied a tightly integrated hybrid architecture. We have extended the basic inference rule of the assertional reasoning mechanism to take directly into account the structure of definitions by altering the meaning of unification. Thus, definitions of terms have a direct, complete, and correct effect on the reasoning process without slowing it down. Further, the term definition language is kept completely distinct from the assertion-making language, and is a more natural one for forming structured definitions and for drawing definitional (analytic) inferences *(e.g.,* subsumption). As implied above, definitions are first-class citiiens in KRYPTON, and are distinct from simple contingent sentences even of similar logical form.

In this section, we describe our alteration to unification in the context of our ABox—Stickel's nonclausal connection graph resolution theorem-prover [Stickel, 1982] [Stickel, 1983] [Stickel, 1985]. This should give the reader a solid idea of how KRYPTON has been made to work, but owing to the brevity of the presentation, it is not intended to be a complete account of the ABox-TBox interface.

A. Connection Graph Theorem-Proving

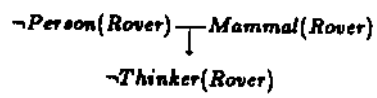
A connection graph (Kowalski, 1975] is a set of wffs and a set of links between the literals[15] of the wffs. For every literal that occurs in a clause input by the user or as the resolvent in a resolution step, there is a corresponding node in the graph. At the time a clause is entered each of its literals is examined and a link is created between each literal and any other literal for which there exists a unifier that causes them to be complementary. Note that there is a crucial distinction between *forming* a link and *resolving upon* a link. A link should be thought of as a marker for identifying two wffs that can be used in a resolution step. The link makes it easier to find the wffs at resolution time, but does not actually perform the resolution. It appears that in general the pre-computation of possible resolution steps allows connection-graph theorem-proving to proceed faster than ordinary resolution [Kowalski, 1975] [Stickel, 1982].

B. Residues

An important piece of information that is stored on a link is the *residue* between the two literals. A residue can be viewed as a statement that the literals connected by a link are contradictory, provided that some additional information, specifically the negation of the residue, is known. When a resolution step is performed, the residue is added as a disjunct to the resolvent in the place of the resolved-away literals. In a conventional unification step, the residue is *"false",* indicating that the literals are directly contradictory; thus a conventional step adds no additional information to the resolvent. However, in KRYPTON we make extensive use of complex non-empty residues for the TBox links.

For example, if we define in the TBox the concept of a *Person* as something that is both a *Mammal* and a *Thinker,* and we know that *Rover* is not a *Person* and that he is a *Mammal,* then by definition *Rover* must not be a *Thinker.* Therefore, the literals - *Person (Rover)* and *Mammalf'Rover)* will be linked together with a residue of *-Thinker(Rover).* This link can be pictured as below, where the literals being linked occur on either side of the horisontal line, and the residue is attached below them:

"A literal it an atomic sentence or the negation of one.



The residue wff is attached only as information associated with the link, but is not actually added to the graph until the link has been resolved upon in a resolution step. So, in this case, none of the links that would connect *-Thinker (Rover)* to other literals are computed until the parent link is resolved upon. The most important consequence of this is that the residue will not be visible for use by the set of support strategy that the ABox employs to direct the progress of a proof (Bee below).

C. Proof Procedure and Resolution Strategy

The proof procedure used with a connection graph is quite simple. The kernel of the procedure is as follows:

1. Stop if the graph contains the empty clause.
2. Select any link from the graph.
3. Generate the resolvent.
4. Construct the new graph:
   (a) Delete the link resolved upon.
   (b) Add links for the resolvent.
5. Return to 1.

Once a given pair of literals in a pair of wffs has been resolved upon, literals that are derived in later resolution steps from the resolved-on literals cannot themselves be resolved upon. This follows from the fact that the original link is deleted *before* links are added between literals in the resolvent and other literals in the graph. As a result of this, connection graph resolution does not allow all the resolution steps that other methods might.

An important point to notice about the proof procedure is that it is nondeterministic. That is, the procedure places no restrictions on how the next link to be resolved upon is chosen. A given implementation must be very careful to embody a strategy that guarantees to preserve completeness of the procedure. One obvious constraint on a suitable strategy should be that the choice of link tend to simplify the resultant graph. Another is that only links that are relevant to the query should be resolved upon.

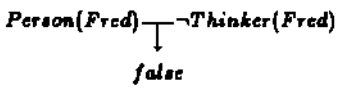
In Stickers theorem prover, these goals are achieved by the use of a complex evaluation function for determining the next link to schedule and by having the evaluation function use set of support as the basic resolution strategy.16 The evaluation function also takes into account such things as how deep the search in a given direction has been and how complicated the residue on the link will be. The hope is that it will choose to ignore links that will result in a more complicated graph and that it will abandon paths that seem to be fruitless. While the evaluation function depends on heuristics and is not guaranteed to find the fastest proof in all cases, our experience shows it to work very well.

D. Terminological Link Types

The ABox is made to understand the TBox by the addition of links to the connection graph that are derived from the terminological definitions in the TBox. Just as in the case of normal connection graph theorem-proving, deduction time is saved by pre-computation of possible unification steps. Further, since our definitions are structured and include a subsumption hierarchy, links constructed from them allow multiple deduction steps *(e.g.,* multiple applications of *modus ponena)* to be accomplished at once. Although we cannot provide evidence of it in the small space we have here (but see [Stickel, 1985]), directly changing the connection graph turns out to be potentially far superior to the mere addition of "meaning postulates" in a loosely integrated system.

Our modification involves the addition of four new connection graph link types. The first three are for concept definitions, and the fourth for role definitions. To a first approximation, the links for a concept correspond to (1) links to concepts that subsume the concept, (2) links to concepts that occur (either explicitly or implicitly) as value restrictions in VRGenerics for the concept, and (3) links to roles that occur in VRGenerics for the concept. Links for the definition of a role are for subchains of that definition that are themselves defined roles. At the very least, this will involve the primitive roles included in the definition.

There are two cases for each of the four basic link types, arising from the necessity and sufficiency of our TBox definitions.17 These are the cases corresponding to positive and negative occurrences of a term. For example, if we know that *Fred* is a *Person*, then we know that his not being a *Thinker* is immediately contradictory, so given the appropriate facts, this link would be necessary:



The *Rover* link of section IV.B shows a link required by the addition of a negative occurrence of a term.

In the next two sections, we will cover in some detail two of the necessary link types arising from concept definitions. The first of these is between a concept and all concepts that subsume it. The second is between a concept and roles that occur in a VRGenerlc in the concept's definition.

In what follows, we shall use these definitions:

- *Grandchild:* (Role Chain *Child Child),* that is, the *Child of* a *Child.*
- *Woman-Student:* (ConGeneric *Woman Student)*, that is, somebody that is both a *Woman* and a *Student.*
- *Successful-Grandma:* (VRGeneric *Woman Grandchild Doctor),* that is, a *Woman* all of whose *Grandchildren* are *Doctors.*

1. Concept-SuperConcept Linking

A concept *P* must be linked to concepts that subsume it. For this concept linking, the TBox must supply the names of all concepts that subsume *P* and also all concepts that it subsumes (information that can be gotten from the TBox using the Knowledge Level operations defined above). The reason for the latter is that we have no way of controlling the order of entry of terms. Finding a concept *Q* that *P* subsumes will allow *Q* to be appropriately linked up when *P* is encountered.

Positive links: Each positive instance of a concept must be linked to all negative instances of concepts that subsume it when the arguments unify in the conventional fashion. No residue is associated with these links. The *Fred* link of the last section is an example of this link type.

16 *Set of support* is a general strategy designed to ensure that only links that point into a "support set" are eligible for selection for the next resolution step. At the beginning of a proof the support set contains only the links between the negation of the query and other wffs. When a link is resohred upon, the old link is removed from the set, and all of the links of the resolvent are added.

17 It should be emphasised that while these link connections may appear complicated, they are not calculated repeatedly. The appropriate links are computed exactly once for each definition, at the time the definition is entered. Therefore the apparent complexity here does not adversely affect the process of deduction.

Negative links: Each negative instance of a concept, *P,* must be linked to all positive instances of concepts, *Q,* that are subsumed by primitive concepts occurring as conjuncts in the definition of *P* when the arguments unify in the conventional fashion. The residue is formed by taking the definition of *P,* removing those pieces of it that are shared with the definition of *Q,* and then negating the result. For example, consider

$$\neg Woman\text{-}Student(Sue) \rightarrow Successful\text{-}Grandma(Sue)$$
$$\neg Student(Sue)$$

If *Sue* is not a *Woman-Student,* but she is a *Successful- Grandma,* then some part of the definition of *Woman-Student* that does not apply to *Successful-Grandma* must not be true for *Sue.* The only piece of *Woman-Student* not included in the other definition is *Student,* so *Sue* must not be a *Student.*

3. Concept-RoleChaln Linking

A concept *P* must be linked to each RoleChaln that occurs in a VRGenerk for it. In order to do this linkage correctly, the TBox must provide the A Box, for each concept, the names of all defined RoleChalns that occur in that concept, and for each defined RoleChaln, the names of all concepts in which it occurs.

Positive links: For every role, *R,* whose definition occurs as a RoleChaln or a subchain of a RoleChaln in some VRGenerk in the definition of *P,* link *P* with the rest of the VRGenerk in which *R* occurs. So, for example, since *Grandchild* is defined as (RoleChaln *Child Child),* *Child* is a defined subchain of *Grandchild.* Therefore, if *Marge* is a *Successful-Grandma* and has a *Child,* then if that *Child* in turn has any *Children,* they must all be *Doctors.*

$$Successful\text{-}Grandma(Marge) \rightarrow Child(Marge, Hope)$$
$$\forall(x)Child(Hope, x) \supset Doctor(x)$$

Negative links: For every role, *R,* whose definition occurs as a subchain of a RoleChaln in some VRGenerk in the definition of P, link *-*P(a)* and *-R(SkFn(a),SkFm(a))* (where *SkFn* and *SkFm* are both Skolem functions associated with the definition of *P).* This link has as its residue the negation of the rest of the definition of *P,* after removing the VRGenerk in which *R* appears.

$$\neg Successful\text{-}Grandma(Charlie) \rightarrow \neg Child(Charlie, x)$$
$$\neg Woman(Charlie)$$

If *Charlie* has no children, then the VRGenerk for *Successful-Grandma* holds vacuously, so for *Charlie* to not be a *Successful-Grandma,* he must fail in the only other requirement, and not be a *Woman.*

3. Other links

In the preceding sections we have outlined two of the major link types in our extension of unification. There are several choices for other links to use to complete the extension. Currently we have implemented one set that, while complete,13 is probably not the best choice, as it yields some unintuitive links to assure that completeness. We have been exploring other link choices that do not of themselves ensure completeness but can be shown to be sufficient when coupled with the set of support strategy. Another approach that is being considered is to not restrict ourselves to binary links, that is links between two literals, but to instead make use of Stickel's more general *theory resolution* capabilities and use n-ary links [Stickel, 1983] [Stickel, 1985]. For example, instead of the link mentioned in the last section concerning *Charlie* which necessitated the addition of a residue, we might have waited until we had all three literals $\neg Successful\text{-}Grandma(Charlie)$, $\neg Child(Charlie, x)$, and $Woman(Charlie)$, and made a single ternary link containing no residue.

18Because of space limitation*, we are not saying here what it means to be complete and consistent with respect to our Knowledge Level specification.

V Example Proof

In this final section, we will present a sample use of KRYPTON. We will supply some TBox definitions and a set of ABox facts and then show how KRYPTON goes about answering a query based on that information.

- TBox definitions:
  Primitive Roles: *Child*
  Primitive Concepts: *Mammal, Thinker, Woman*
  Defined Concepts:
  Person (ConGeneric *Mammal Thinker*)
  NoSon (VRGeneric *Person Child Woman*)
- ABox facts:
  $Child(Fred, Pat)$
  $Child(Mary, Sandy)$
  $NoSon(Fred) \vee NoSon(Mary)$

With the facts above, we should be able to show that there is somebody in the world who is a *Person* and has a *Child* that is a *Woman,* even though we do not know who that somebody is. This query is formulated as $\exists x \exists y[Person(x) \wedge Child(x, y) \wedge Woman(y)]$. The intuition behind the proof is that if *Fred* and *Pat* both have children and at least one of them is a *NoSon,* then whichever of them it is is himself a *Person* and has a *Child* that is a *Woman* (given the definition of *NoSon*). That either *Fred* or *Mary* is a *NoSon* is insufficient information for this proof, since the definition of *NoSon* does not require that such a person have a *Child,* merely that if she has a *Child,* then that *Child* is a *Woman.* Note that any proof of this is going to have to use terminological information to know, for example, that the *Child* of whichever of them is a *NoSon* is also a *Woman.*

The proof proceeds by trying to derive a contradiction from the known facts and the negation of the query. Lines 1-3 are the known ABox facts that will be used in the proof. Line 4 is the negation of the query. x and y are used as universal variables.

1. $Child(Fred, Pat)$
2. $Child(Mary, Sandy)$
3. $NoSon(Fred) \vee NoSon(Mary)$
4. $\neg Person(x) \vee \neg Child(x, y) \vee \neg Woman(y)$
5. $\neg Person(Fred) \vee \neg Woman(Pat)$

Normal resolution on 1 and $\neg Child(x, y)$ in 4.

6. $\neg Person(Fred) \vee NoSon(Mary) \vee \neg Child(Fred, Pat)$

By 3, *Fred* is possibly a *NoSon,* which means that all his *Children* are *Women* (from the terminology). Stating that *Pat* is not a *Woman* in 5 has the consequence that *Pat* cannot be *Fred's Child.* In other words, $NoSon(Fred)$ in 3 and $\neg Woman(Pat)$ in 5 resolve away and leave a residue of $\neg Child(Fred, Pat)$.

7. $\neg Person(Fred) \vee NoSon(Mary)$

Normal resolution on 1 and $\neg Child(Fred, Pat)$ in 6.

8. $NoSon(Mary)$

By the definition of *NoSon,* if *Fred* is one then he must also be a *Person,* so $\neg Person(Fred)$ in 7 and $NoSon(Fred)$ in 3 are directly contradictory.

9. $\neg Child(Mary, y) \vee \neg Woman(y)$

This time, if *Mary* is a *NoSon,* she must be a *Person,* so 8 and

-Person(x) in 4 are directly contradictory, with *Mary* being substituted for z in the resolvent.

10. *-Child(Mary,y)*

If *Mary* is a *NoSon* (as stated in 8), any *Children* she might have must be *Women.* Therefore, if there are no *Women* at all (as stated in 9), then *Mary* must have no *Children.* In this case, the residue, *-Child(Mary,y),* was already part of the resolvent of 8 and 9, so it does not need to be added again.

11. *false*

Normal resolution on 10 and 2.

VI Conclusions

As we have shown, KRYPTON is a tightly integrated hybrid reasoning system that provides both terminological and assertional facilities. At the Knowledge Level, it can be seen that assertional reasoning takes into account the definitions of terms expressed in a special-purpose frame-based description language. At the Symbol Level, it can be seen that this interface is implemented by augmenting a theorem-prover's notion of unification to accommodate definitional relationships between predicates. Our KRYPTON implementation currently runs on a Symbolics 3600.

In some respects, this resembles what Stickel has called "theory resolution" [Stickel, 1983] [Stickel, 1985], wherein non-equational theories can be built directly into the theorem-prover. One could—at the Symbol Level—accurately call KRYPTON an implementation of partial theory resolution, in that we have used the TBox to implement specialized reasoning procedures for certain tasks. However, as discussed in Section III, we have a full formal semantics at the Knowledge Level that supports the first-class citisenship of descriptive terms in KRYPTON. Theory resolution allows us a convenient implementation of this idea, but does not itself provide us with the suitable semantics.

KRYPTON is different, in fact, from most other hybrid approaches in that it directly, and soundly, integrates two fundamentally different kinds of representation and reasoning.

References

[Anderson and Belnap, 1975] Anderson, A. R., and Belnap, N. D., Jr., *Entailment: The Logic of Relevance and Necessity.* Vol. 1. Princeton, NJ: Princeton University Press, 1975.

[Brachman, 1985] Brachman, R. J., "'I Lied About the Trees.'" *AI Magazine,* Vol. 6, No. 3, Fall, 1985.

[Brachman and Levesque, 1982] Brachman, R. J., and Levesque, H. J., "Competence in Knowledge Representation ," in *Proc. AAAI-82,* Pittsburgh, August, 1982, 18&-192.

[Brachman and Levesque, 1984a] Brachman, R. J., and Levesque, H. J., "The Tractability of Subsumption in Frame-Based Description Languages," in *Proc. AAAI-84,* Austin, TX, August, 1984, 34-37.

[Brachman and Levesque, 1984b] Brachman, R. J., and Levesque, H. J., "What Makes a Knowledge Base Knowledgeable? A View of Databases from the Knowledge Level," in *Proc. First International Workshop on Expert Database Systems,* Kiawah III., S. C, October, 1984, 30-39.

[Brachman and Schmolse, 1985] Brachman, R. J., and Schmolse, J. G., "An Overview of the KL-ONE Knowledge Representation System." *Cognitive Science,* 9(2), April-June, 1985, 171-216.

[Brachman, *et al,* 1983a] Brachman, R. J., Fikes, R. E., and Levesque, H. J., "Krypton: Integrating Terminology and Assertion," in *Proc. AAAI-88,* Washington, DC, August, 1983, 31-35.

[Brachman, *et al.,* 1983b] Brachman, R. J., Fikes, R. E., and Levesque, H. J. "Krypton: A Functional Approach to Knowledge Representation." *IEEE Computer, Special Issue on Knowledge Representation,* October, 1983, 67-73.

[Brown and Burton, 1975] Brown, J. S., and Burton, R. R., "Multiple Representations of Knowledge for Tutorial Reasoning," in *Representation and Understanding: Studies in Cognitive Science.* D. G. Bobrow and A. Collins, *eds.* New York: Academic Press, 1975, 311-349.

[Genesereth, 1981] Genesereth, M., "The Architecture of a Multiple Representation System," Memo HPP-81-6, Comp. Sci. Dept., Stanford Univ., May, 1981.

[Kowalski, 1975] Kowalski, R., "A Proof Procedure Using Connection Graphs", *JACM,* 22(4), October, 1975, 572-695.

[Levesque, 1984a] Levesque, H. J., "A Fundamental Tradeoff in Knowledge Representation and Reasoning," in *Proc. CSCSI84,* London, Ontario, May, 1984, 141-152.

[Levesque, 1984b] Levesque, H. J., "A Logic of Implicit and Explicit Belief," in *Proc. AAAI-84,* Austin, TX, August, 1984, 198-202.

[Levesque, 1984c] Levesque, H. J., "Foundations of a Functional Approach to Knowledge Representation." *Artificial Intelligence,* 23(2), July, 1984, 155-212.

[Moses, 1971] Moses, J., "Algebraic Simplification: A Guide for the Perplexed." *CACM,* 14(8), August, 1971, 527-537.

[Newell, 1981] Newell, A., "The Knowledge Level," *AI Magazine,* 2(2), Summer, 1981, 1-20.

[Patel-Schneider, 1985] Patel-Schneider, P. F., "A Decidable First-Order Logic for Knowledge Representation," *Proc. IJCAI-85.*

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

[Sloman, 1971] Sloman, A., "Interactions Between Philosophy and Artificial Intelligence: The Role of Intuition and Non-Logical Reasoning in Intelligence." *Artificial Intelligence,* 2 (1971), 209-225.

[Sloman, 1984] Sloman, A., "Why We Need Many Knowledge Representation Formalisms," British Computer Society Expert Systems Conference, December, 1984.

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

[Stickel, 1983] Stickel, M. E., "Theory Resolution: Building in Nonequational Theories," in *Proc. AAAI-88,* Washington, DC, August, 1983, 391-397.

[Stickel, 1985] Stickel, M. E., "Automated Deduction by Theory Resolution," to appear in *J. Automated Reasoning,* 1985.

[Vilain, 1985] Vilain, M., "The Restricted Language Architecture of a Hybrid Representation System," *Proc. IJCAI-85.*