THE INTERACTION WITH INCOMPLETE KNOWLEDGE BASES: A FORMAL TREATMENT*

Hector J. Levesque

Department of Computer Science
University of Toronto
Toronto, Ontario

ABSTRACT

Some formal representation Issues underlying the Interaction between an expert system and Its knowledge base are discussed. It Is argued that a language that can refer both to the application domain and to the state of the knowledge base Is required to specify and to question an incomplete knowledge base. A formal logical language with this ability is presented and its semantics and proof theory are defined. It is then shown how this language must be used to interact with the knowledge base.

1. INTRODUCTION

An Important characteristic of any knowledge-based system Is its interaction with a knowledge base (or KB for short) that provides and maintains information about the application domain. In general, there are two distinct modes of interaction between an expert system and its KB: the system will want to ask and to tell the KB about the application area. In this paper, I will examine what special expressive requirements are placed on the language(s) of interaction when a system must deal with a KB that is Incomplete.

Loosely speaking* a KB is Incomplete when it does not have all the Information necessary to answer some question of Interest to the system. When a knowledge-based system is forced to depend on an incomplete KB, its ability to make decisions or solve problems is seriously compromised. In some cases, the lack of knowledge can be circumvented using general defaults [1], while in other situations, special heuristics are required [2]. However, no matter how a system plans to deal with incompleteness, it must first be able to determine where this incompleteness lies. In other words, a system has to find out exactly where knowledge Is lacking before it can decide what to do about it. This suggests that a KB must be capable of providing information not only about the application area, but about Itself as well. Thus, the language used to interact with a KB must allow a user to define and inquire about what the KB does and does not know. The major issue of this paper, then, is what impact this capability will have on the language of interaction.

This work was sponsored in part by the Natural Sciences and Engineering Council of Canada and the Department of Computer Science, U. of Toronto.

The approach taken in my work [3] focuses primarily on the formal aspects of this issue. As described in section two, I will be dealing with KB's consisting of formulas of the first order predicate calculus. In section three, I examine what questions one would like to be able to ask such a KB and conclude that these questions are best phrased in a superset of the predicate calculus I call KL. In section four, I examine the language KL in detail and provide a semantics and proof theory. Given this analysis of KL, I then consider its use as a language to interact with a KB: in section five, the formulas of KL are used to query a KB and I show that this involves a non-monotonic version of the language; in section six, the formulas of KL are used to define a KB and I indicate why this requires converting the formulas into the language of the KB.

Three areas of related research should be noted. First of all, my formalization of KL owes much to the work on the logic of knowledge and belief pioneered in [4] and continued in (5] and [6] (though my application is somewhat different). Secondly, there has been research on incompleteness from the database area ([7],[8]) concentrating primarily on efficient query evaluation. Finally, although the research here does not deal with defaults or assumptions, the work on non-monotonic reasoning ([9],(10], [11], [12]) has provided technical inspiration.

2. FIRST ORDER KNOWLEDGE BASES

As described in [13], a first order KB is a finite set of closed formulas of first order logic (FOL). I will restrict my attention to consistent sets of formulas that do not contain function symbols other than a (possibly infinite) set of O-ary symbols called constants. Moreover, a bijection is assumed to exist between the set C of constants and the set D of entitles in the domain of discourse.

A query in this framework is any formula of FOL and is answered by consulting the provability relation H to determine what does and does not follow from the KB. For each closed query there are three possible replies: yes, no or unknown, depending on whether the query, its negation or neither follow from what is available in the KB. A KB is incomplete when some query has an unknown answer and complete otherwise.

Consider a typical KB containing facts such as

Teachea(John, bill), Student(bill),...

To avoid having to include within this KB all the "negative" facts about the application such as

-Student (John), nTeaches(bill,john),.••

an assumption is usually made (once and for all) that the negation of any atomic formula can be inferred from the inability to infer the atomic formula. This is the closed world assumption (CWA) [14] which, along with the assumption that there is a bljection between constants and entitles in the domain, guarantees that any KB is complete.

If, on the other hand, we have a KB that contains

[Student(mary) v Student(susan)]

but does not contain either

Student(mary) or Student(susan),

the CWA cannot be used since it would allow us to conclude that neither Mary nor Susan are students, contradicting what is in the KB. This KB only partially describes a world, in that it specifies that one of Mary or Susan is a student but not which. Thus, there is a query that is true in the intended application but whose truth cannot be determined on the basis of what is available in the KB. The KB is, consequently, incomplete.

For the rest of this paper, I will say that a KB knows a formula if it answers yes to the query and, for any predicate p and constant c, that c is a known p when the formula p(c) is known.

3. THE KNOWLEDGE BASE QUERY LANGUAGE

Even though a KB is incomplete, there may be completeness in certain areas. For example, suppose that

KB (x)[Teacher(x) = (x-John v x-Jim)]

In this case, not only are John and Jim teachers, but the KB knows that they are the only teachers. It therefore has a complete picture of the teachers. On the other hand, it may be the case that

KB h (Ex)Teacher(x)

without there being any known teachers. In this situation, the KB knows that there is a teacher but does not know who and so does not have a complete list of teachers. There is, moreover, a third possibility which is that it cannot be determined from what is available in the KB whether or not it has complete knowledge of the teachers. So just as the question

Is John a teacher?

may be answered yes, no or unknown, the question

Are all the teachers known?

may also be yes, no or unknown depending only on what is in the KB.

If the purpose of a query language is to provide an accurate picture of what is and is not available in the KB, we should be able to formulate queries that ask the KB about its incompleteness. In FOL, a query that asks if there is a teacher apart from the known teachers might be expressed by

(Ex)[Teacher(x) -»(x«cl V...V x«ck)]

where the c^{f}s are constants ranging over all the teachers known to the KB.

There are a couple of problems with this method of asking if all the teachers are known. First of all, there could be a very large number of known teachers. In some applications and for some predicates, there may even be an Infinite number. Secondly, we have to know what these constants are in the first place to be able to formulate the query. For a large and complex KB, it could happen that only the KB has this information. One can also Imagine situations where the KB will not divulge this information for security reasons while still being willing and able to answer questions about its incompleteness.

This suggests that the KB Itself should keep track of Its Incompleteness in the same way it maintains knowledge of the application area. One possibility, for example, Is to have a predicate "Known-teacher" and to allow a KB containing the following:

Teacher(john), Known-teacher(John),
[Teacher(Jim) v Teacher(dan)],
-Known-teacher (jim), -Known-teacher (dan).

The problem with this approach (and any other that Involves a direct encoding into FOL) is the management of this extended language. There is a very definite relationship between "Teacher" and "Known-teacher" that must be captured somehow. Among other things, we want that whenever

Teacher(c)

is in the KB, then

Known-teacher(c)

is as well. The closest we can come to expressing this is by an axiom stating that

(x)[Teacher(x) = Known-teacher(x)].

However, this does not work since this formula is inconsistent with the above KB. The correspondence between "Teacher" and "Known-teacher" is much more complex. In a nutshell, "Known-teacher" should not be a predicate for the simple reason that Its truth or falsity does not depend on the domain being modelled but on the model (i.e. the KB) Itself.

The solution, then, is to leave the KB as is

but to extend the query language to allow questions that refer to the current state of knowledge of the KB. The query language I propose, called KL, contains all of FOL and, in addition, has formulas of the form

Ka

read as

The KB knows that a.

This leaves us with a first order KB while still allowing us to query the KB regarding its incompleteness. In particular, we have a new form of query evaluation

$KB \vdash \alpha$

where $\vdash$ is some (as yet to be specified) provability relation and a is any formula of KL. For example, to find out if the KB has an Incomplete list of teachers, we ask if

$KB \vdash (Ex)[Teacher(x) \wedge \neg K[Teacher(x)]]$.

Similarly, while the query

(Ex)Teacher(x)

can be used to find out if there are any teachers, the query

$(Ex)K[Teacher(x)]$

asks if the KB knows who any of them are. To be able to define the $\vdash$ relation, we must first look at the semantics of the language KL Itself.

4. THE LANGUAGE KL

The query language KL has the same formation rules as FOL but also includes the rule

If $\alpha \in KL$ then $K\alpha \in KL$.

There are, consequently, two kinds of formulas in KL: the first, like

$p(c) \wedge (Ex)q(x)$

will be true or false depending only on how the world is, that is, on the interpretation of the constant and predicate symbols; the second, like

$K[p(c) \wedge (Ex)q(x)]$

will be true or false depending only on how the KB is, that is, on what is and is not known. I will call the latter formulas **pure**. There are also formulas in KL that are mixtures of the two types and whose truth value depends both on the world and the KB. KL also allows for **meta-knowledge** in formulas such as

K[(Ex)Kp(x)]

which talk about the KB's knowledge of its own knowledge.

The semantic Interpretation of a closed formula of KL will depend on both a world description (or interpretation, in the Tarsklan sense) and a description of a KB. In general, a KB can be viewed as a partial description of s world and can thus be characterised by a set of world descriptions. To make all of this more precise, first note that because of the assumed bljection between constants snd entitles in the domain, a world description need only assign a truth value to the atomic sentences. Thus, we csn define the set of world descriptions as

$WD = [ATOMS \to \{T,F\}]$.

KB descriptions are, then, Just non-empty sets of world descriptions:

$MD = \{m \subseteq WD \mid m \text{ not empty}\}$.

The interpretation of any closed formula a is provided by the function V.

$V \in [KL \times WD \times MD \to \{T,F\}]$ is defined by
$V(\alpha,w,m) = w(\alpha)$ when α is atomic,
$V(\neg\alpha,w,m) = T$ iff $V(\alpha,w,m) = F$,
$V(\alpha \vee \rho,w,m) = T$ iff $V(\alpha,w,m) = T$ or $V(\rho,w,m) = T$,
$V((Ex)\alpha,w,m) = T$ iff for some c, $V(\alpha<x/c>,w,m) = T$,
$V(K\alpha,w,m) = T$ iff for every $w' \in m$, $V(\alpha,w',m) = T$.

A formula is valid when it is true with respect to every world and KB description. In the case where a is pure, I will use V(a,m) to refer to the truth value of a with respect to the KB described by m.

Turning now to the proof theory for KL, since the language includes FOL as a subset, we will need the axioms of FOL and its two inference rules: modus ponens snd universal generalization. To account for the K operator, we have to realize that it behaves like s provability relation in that something is known when it "follows" from what is available in the KB. We will, therefore, insist that the axioms of FOL are known and that the KB Is able to perform modus ponens and universal generalization based on what is known. This might be called the assumption of competence. As for meta-knowledge, it is convenient to assume that the KB knows the correct truth value of any pure formula. In other words, there is never any reason to tell the KB about itself nor is there any reason to doubt what the KB knows about Itself. No matter how incomplete or inaccurate a KB can be about the world, it is assumed to be the final authority on itself. This might be called the assumption of closure. Note that this assumption applies only to pure sentences. Fortunately, these can be given a syntactic characterization: a formula is pure iff every occurrence of a predicate symbol appears within the scope of a K operator. We therefore have the following axlomatization of the language KL:

Axiom Schemata

1. The axioms of FOL
2. $K\alpha$ where α is an axiom of FOL
3. $K(\alpha \supset \rho) \supset (K\alpha \supset K\rho)$
4. $(x)K\alpha \supset K(x)\alpha$
5. $\alpha \equiv K\alpha$ where α is pure

Rules of Inference

Modus Ponens and Universal Generalization

If we let $\vdash$ denote the provability relation for this axiomatization, then the key result here is that the proof theory is both sound and complete with respect to the semantics given above:

Theorem: $\vdash \alpha$ iff α is valid.

One thing to notice is that by the closure property, we have that for any pure a

$$\vdash K\alpha \supset \alpha.$$

This is, however, not the case for arbitrary a in that, for example, the set

$$\{\neg p(c), Kp(c)\}$$

is satisflable and, hence, logically consistent. This is a situation where the KB is behaving properly but just happens to be mistaken about the world. So, in some sense, the K operator should be read as "believe" rather than "know". On the other hand, the kind of belief Involved here is very special in that a KB will always think it is dealing with knowledge since

$$\vdash K(K\alpha \supset \alpha)$$

for any formula a. So there is an aspect of commitment to what is believed in that a KB will never believe it has mistaken beliefs.

5. QUERIES REVISITED

Having examined the semantics and proof theory of KL, we are still faced with the problem of specifying what is meant by

$$KB \mid\!\!\sim \alpha \text{ where } KB \subseteq FOL \text{ and } \alpha \in KL.$$

The idea here is that this should mean

"If I know only what is in KB, then I know α."

Semantlcally, this is saying that Ka is true with respect to the knowledge base that has the least amount of world knowledge consistent with knowing everything in KB. Thinking of a knowledge base as described by a set of world descriptions, we want the least amount of world knowledge and, thus, the largest possible set.

$$\text{Let } M(KB) = \{w \mid w \text{ satisfies the KB}\}.$$

Any knowledge base that knows more than what is in the KB is described by a subset of M(JCB) and, so, has a more refined view of the world. This suggests how query evaluation can be defined semantlcally:

$$\text{Let } KB \mid\!\!\sim \alpha \text{ iff } V(K\alpha, M(KB)) = T.$$

So, the answer to a question is yes exactly when it is known to be true in M(KB). For example, if

$$KB1 = \{Teacher(john)\}$$

then

$$KB1 \mid\!\!\sim Teacher(john),$$
$$KB1 \mid\!\!\sim \neg K[Student(bill)],$$
$$KB1 \mid\!\!\sim (Ex)K[Teacher(x)],$$
$$KB1 \mid\!\!\sim K\neg K[Student(bill)],$$
$$KB1 \mid\!\!\sim \neg K[(Ex)[Teacher(x) \wedge \neg KTeacher(x)]].$$

The last statement above confirms that KB1 knows that it does not know if it has a complete list of teachers. Note that 11- is a non-monotonlc operator in that it is not the case that

$$KB1 + Student(bill) \mid\!\!\sim \neg K[Student(bill)].$$

However, unlike a fully general non-monotonlc logic as presented in [9] or [10], the $\mid\!\!\sim$ operator here has been given a simple and natural semantic characterization. Below I will present a proof theoretic analogue of this operator and claim a soundness and completeness type result, but before doing so it is necessary to examine how KL can be used to specify a KB.

6. THE KNOWLEDGE BASE DEFINITION LANGUAGE

Since, for incomplete KB's, the CWA is not used uniformly, It would be extremely convenient to be able to tell the KB when (if ever) the assumption could be used for special cases. Conversely, we should also be able to tell the KB when the CWA cannot be used. If we let

$$\lambda = (Ex)[Teacher(x) \wedge \neg KTeacher(x)]$$

then $\neg\lambda$ states that if someone is not currently known to be a teacher then he is not a teacher. So $\neg\lambda$ is the CWA relativized to teachers while λ itself is a statement that this assumption cannot be used (because there are teachers other than the currently known ones). The question immediately arises as to whether or not we can add formulas such as λ or $\neg\lambda$ to a KB, thus generalizing a KB to be any consistent set of formulas from KL instead of FOL.

There are a number of problems with this generalization but I will address only one of these relating to the formula λ. The idea here is that we would start off with a KB such as KB1 defined earlier. Since KB1 does not know whether or not John is the only teacher, it is not the case that either

$$KB1 \mid\!\!\sim \lambda \quad \text{or} \quad KB1 \mid\!\!\sim \neg\lambda.$$

Suppose we consider telling it that it does not have all the teachers and get

$$KB2 = KB1 + \lambda.$$

If we now want to tell the KB that Jim is a teacher, we get

KB3 - KB2 + Teacher(Jim).

The problem here is that KB3 still contains λ end, consequently, still thinks it is missing a teacher. In fact, no matter how many teachers we tell the KB about, it will still think it is missing at least one. Moreover, if we try to tell it that it finally has all of them by adding $\neg\lambda$, then we arrive at an Inconsistent KB since it also contains λ.

What should have happened here is somewhat different. Once we arrived at KB2, the KB should know that it is missing a teacher and hence believe λ. However, once a new teacher is added to the KB to produce KB3, the KB has no way of knowing whether or not this is the last teacher it was missing and so it should be the case (as with KB1) that neither

$$KB3 \mid\!\!- \lambda \quad \text{nor} \quad KB3 \mid\!\!- \neg\lambda.$$

In other words, after the introduction of Jim, the KB should no longer know whether or not it has all the teachers. In fact, the knowledge it had as KB2 is lost when it becomes KB3.

This is a strange kind of non-monotonicity. The usual symptom of a non-monotonic logic is that the addition of a new axiom Invalidates a previous theorem. In our case, the addition of the new axiom Invalidated a previous axiom. The curious puzzle here is that there is no "belief revision" going on in the sense of a realisation that an axiom was incorrectly added to the KB. Similarly, there is no admission of the world having changed in the sense of someone becoming or ceasing to be a teacher. In fact, the only change that has taken place is a change in what is known about the world. But this is enough since λ does make reference to the current state of the KB. So without admitting that the world has changed or that some previous statement about the world needs revision, we can still maintain that the truth value of λ can change by noting that the state of the KB has changed. Thus, λ cannot be part of the KB since the state of knowledge it refers to disappears as Information is acquired.

The solution to the problem of the addition of λ to a KB is, therefore, to treat the formula as ordinary world knowledge where the K operator is used to refer to the current state of the KB. This is only natural since a KB la assumed to have complete knowledge of itself. Conaequently, any mention of what is known in a new piece of information must be "referential" and not "attributive". Thus, the addition to the KB must be understood by first resolving these references. In other words, the solution to the problem is not to prohibit additions like λ, but rather to allow λ in the KB definition language but not the KB itself. For example, KB2 now becomes

$$KB2 = KB1 + \lambda = \{Teacher(john), (Ex)[Teacher(x) \wedge \neg(x=john)]\}$$

where we have replaced the open formula

KfTeacher(x)]

by a first order formula that resolves this reference with respect to KB1:

(x-john).

This produces the property that

$$KB2 \mid\!\!- \lambda$$

and neither KB3 11 A nor KB3 |(- -,A as desired.

The solution I have proposed above presupposes that there will always be a formula of FOL that can be used to resolve any reference to what Is known. It Is worth noting that this cannot be done Independently of a KB In that there Is no formula α of FOL such that

$$\vdash K\lambda \equiv K\alpha.$$

Of course, all we really need Is a formula for each KB and not a formula that works for every KB. Fortunately, If we assume the availability of an equality predicate and restrict ourselves to finite KB's, this can always be done.

Theorem: Assume KB is finite and $\alpha(x1,\ldots,xk) \in KL$. There is a formula $/\alpha/ \in FOL$ (with equality) such that $KB \mid\!\!- (x1)\ldots(xk)[K\alpha \equiv K/\alpha/]$.

The formula required here can be defined by:

$/\alpha/ = \alpha$ when $\alpha \in FOL$, $/\neg\alpha/ = \neg/\alpha/$, $/(\alpha\supset\rho)/ = (/\alpha/\supset/\rho/)$, $/(x)\alpha/ = (x)/\alpha/$ and $/K\alpha/ = RESOLVE[/\alpha/]$.

RESOLVE[α] = If α has no free variables
then if $\overline{KB} \vdash \alpha$ then $(x)(x=x)$ else $\neg(x)(x=x)$
/* Assume x is free in α and that c1,...,ck but not c are all the constants in α or KB */
else $[(x=c1 \wedge RESOLVE[\alpha<x/c1>]) \vee \ldots \vee (x=ck \wedge RESOLVE[\alpha<x/ck>]) \vee (x\neq c1 \wedge \ldots \wedge x\neq ck \wedge RESOLVE[\alpha<x/c>]<c/x>)]$

The method of allowing all of KL to specify a KB is thus to let

$$KB + \alpha = KB \cup \{/\alpha/\} \quad \text{for any } \alpha \in KL.$$

Viewed more semantics Uy In terms of world descriptions, we have that

$$M(KB) \cap \{w \mid V(\alpha,w,M(KB))=T\} = M(KB \cup \{/\alpha/\}).$$

Note that although the function RESOLVE is not recursive, It is strictly proof-theoretic. Moreover, it can be shown that

Theorem: For any $\alpha \in KL$, $KB \mid\!\!- \alpha$ iff $KB \vdash /\alpha/$.

which defines a syntactic version of query evaluation that is exactly equivalent to the semantic one defined earlier. Of course, the proof theory is not axiomatic but this is to be expected given that

$$\{\alpha \in KL \mid \mid\!\!- \alpha\}$$

is not recursively enumerable. So, to summarise,

although the interaction with a first order KB should allow the language KL to be used, this Interaction can be understood in first order terms when the KB is finite.

7. CONCLUSION

In this paper, I have considered from a formal standpoint the problem of interacting with an incomplete KB. The motivation behind the research is that, to effectively deal with partial knowledge, a system must first be able to determine the exact limits to what is known. This, In turn, places certain requirements on the language used to interact with a KB. In particular, the language has to be able to refer to the state of the KB as well as to the state of the application domain.

To this effect, I proposed a language KL for interacting with a first order KB and presented a proof theory and semantics which were direct extensions of their FOL counterparts. KL was then applied to query evaluation and I showed that this required a non-monotonic operator. Semantic and syntactic interpretations of this operator were provided. Finally, KL was applied to KB definition and I demonstrated why this required reducing KL into the language of the KB, FOL. In fact, the net result was that the KB remained first order, but that interaction with it took place In a more expressive language.

Apart from the more practical implications of this work, there remain open questions even within the formal framework. I have not mentioned, for example, what impact the presence of an equality relation or non-constant terms would have on KL. Also, a method of handling defaults (and exceptions) is a reasonable goal given that the framework allows one to determine where their application is needed. In summary, the framework provides not only a formal standard against which to measure representation languages, but also a basis for further exploration.

REFERENCES

[1] Reiter, R., "On Reasoning by Default", Proc. second TINLAP conference, Urbana, Illinois, 1978.

[2] Collins, A., Warnock, E., Alello, N. and Miller, M., "Reasoning from Incomplete Knowledge", Representation and Understanding: Studies in Cognitive Science, Collins, A. and Bobrow, D. (eds.), Academic Press, 1975.

[3] Levesque, H., "A Formal Treatment of Incomplete Knowledge Bases", Ph.D. thesis, Dept. of Computer Science, U. of Toronto, 1981.

[4] Hintikka, J., Knowledge and Belief; An Introduction to the Logic of the Two Notions. Cornell University Press, 1962.

[5] Moore, R., "Reasoning about Knowledge and Action", Ph.D. thesis, Dept. of E.E. and Computer Science, MIT, 1980.

[6] McCarthy, J., Sato, M., Hayashi, T. and Igarashl, S., "On the Model Theory of Knowledge", Memo AIM-312, Dept. of Computer Science, Stanford U., 1978.

[7] Vassiliou, Y., "A Formal Treatment of Imperfect Information in Database Management", Ph.D. thesis, Dept. of Computer Science, U. of Toronto, 1980.

[8] Llp8kl, V., "On Semantic Issues Connected with Incomplete Information Data Bases", Institute of Computer Science PAS Report 325, Warsaw, Poland, 1978.

[9] McDermott, D. and Doyle, J., "Non-monotonic Logic", AI memo 486, AI Lab., MIT, 1978.

[10] McDermott, D., "Non-monotonic Logic II: Non-monotonic Modal Theories", Research Rep. No. 174, Dept. of Computer Science, Yale U., 1980.

[11] McCarthy, J., "Circumscription - A Form of Non-monotonic Reasoning", Artificial Intelligence 13, 1980.

[12] Reiter, R., "A Logic for Default Reasoning", Artificial Intelligence 13. 1980.

[13] Gallaire, H. and Minker, J. (eds.), Logic and Data Bases. Plenum Press, New York, 1978.

[14] Reiter, R., "On Closed World Data Bases", Logic and Data Bases, Gallaire, H. and Minker, J. (eds.), Plenum Press, 1978

ACKNOWLEDGEMENTS

I would like to thank John Mylopoulos and Alex Borglda for contributing to the development of this work. I am also grateful to Teresa Miao for typing this manuscript.