Foundations of a Functional Approach to Knowledge Representation

Hector J. Levesque
Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA 94304, U.S.A.

Recommended by Terry Winograd

ABSTRACT

We present a new approach to knowledge representation where knowledge bases are characterized not in terms of the structures they use to represent knowledge, but functionally, in terms of what they can be asked or told about some domain. Starting with a representation system that can be asked questions and told facts in a full first-order logical language, we then define ask- and tell-operations over an extended language that can refer not only to the domain but to what the knowledge base knows about that domain. The major technical result is that the resulting knowledge, which now includes auto-epistemic aspects, can still be represented symbolically in first-order terms. We also consider extensions to the framework such as defaults and definitional facilities. The overall result is a formal foundation for knowledge representation which, in accordance with current principles of software design, cleanly separates functionality from implementation structure.

0. Introduction

Traditionally, the goal of knowledge representation systems has been to provide flexible ways of creating, modifying, and accessing collections of symbolic structures forming knowledge bases. In this paper, we present a new view of knowledge bases and their role in knowledge-based systems. Essentially, a knowledge base (or KB, for short) is treated as an *abstract data type* that interacts with a user or system only through a small set of operations. In our case, we consider only two interaction operations: one allows a system to ask (ASK) the KB questions about some application domain and the other allows the system to tell (TELL) the KB about that domain. The complete functionality of a KB is measured in terms of these operations; the actual mechanisms and structures it uses to maintain an evolving model of the domain are its own concern and not accessible to the rest of the knowledge-based system. In Section 1, we elaborate on this theme in terms of Newell's concept of a knowledge level.

Artificial Intelligence **23** (1984) 155–212
0004-3702/84/$3.00 © 1984, Elsevier Science Publishers B.V. (North-Holland)

From our point of view, the capabilities of a KB are strictly a function of the range of questions it can answer and assertions it can accept. In Section 2, we examine a language that can be used for questions and assertions. Specifically, we investigate in detail the semantics and proof theory of an interaction language $\mathscr{L}$ that is a dialect of first-order logic. We then show some of its limitations as an interaction language and define a new one called $\mathscr{KL}$ that greatly extends what a KB can be told or asked. This language will have the ability to refer both to the application domain and to what the KB *knows* about that domain. The operations of TELL and ASK (and consequently, the functionality of a KB) are defined in a representation-independent way in terms of this language.

In a sense, the language $\mathscr{KL}$ is the core of our functional view of a KB. First of all, the two interaction operations are defined directly in terms of the constructs it provides. But it also characterizes a KB more directly: the language contains a primitive operator $\boldsymbol{K}$ which can be read as 'the KB currently knows that . . .'. As such, the valid sentences of $\mathscr{KL}$ specify what must be known or not known by a KB of the kind we are considering.

In Section 3, we examine how to represent knowledge symbolically in order to realize the functionality required by the TELL and ASK operations. In particular, we present a proof that (under a few reasonable assumptions) the required knowledge can be represented in first-order terms even though $\mathscr{KL}$ is not a first-order language. This Representation Theorem in fact proves that a first-order implementation of a KB is *correct* in that it meets its specification (in terms of the definition of the operations).

Finally, in Section 4, we show how the framework we have presented can be extended and applied to some of the ongoing research topics in knowledge representation. Among other things, we examine how defaults (and non-monotonic reasoning) can be handled and how a 'hybrid' system containing aspects of both logical and object-oriented representation paradigms might be accomodated.

1. The Knowledge Level

In a recent paper [1], Newell introduces the idea of a *knowledge level*, an abstraction in terms of which intelligent agents can be described. For the rest of this section (and much of the paper), we will show how this idea applies equally well to knowledge bases and indeed motivates our functional approach.

1.1. Competence

A major characteristic of the knowledge level is that knowledge is considered to be a *competence* notion, "being a potential for generating action".[1] There

[1]All quotations in this section are from [1].

are actually two ways of looking at this. First, we might want to say that, to a first approximation, if an agent believes p and if p entails q, then he is likely to believe q. In this case, competence might be thought of as a plausible abstraction (or heuristic) for reasoning about the beliefs of an agent. But there is another way of looking at this: we can say that if an agent imagines the world to be one where p is true and if p entails q, then (whether or not he realizes it) he imagines the world to be one where q also happens to be true. In other words, if the world the agent believes in satisfies p, then it must also satisfy q. The point is that the notion of competence need not be one where an agent is taken to have appreciated the consequences of what he knows, but merely one where *we* examine those consequences.

So the analysis of knowledge at the knowledge level is really the study of what is *implicit* in what an agent might believe[2], precisely the domain of *logic*. As Newell points out, "just as talking of *programmerless* programming violates truth in packaging, so does talking of a *non-logical analysis* of knowledge". This is not to imply that agents do, can, or should use some form of logical calculus as a representation scheme, but only that the analysis of what is being represented is best carried out in this framework.

But what does this analysis amount to? Are there any purely logical problems to be solved? Indeed, once what Newell calls the *symbol level* has been factored out, is there *anything* concrete left to say? Newell admits the following:

> However, in terms of structure, a body of knowledge is extremely simple compared to a memory defined at lower computer system levels. There are no structural constraints to the knowledge in a body, either in capacity (i.e., the amount of knowledge) or in how the knowledge is held in the body. Indeed, there is no notion of how knowledge is held (*encoding* is a notion at the symbol level, not knowledge level). Also, there are not well-defined structural properties associated with access and augmentation. Thus, it seems preferable to avoid calling the body of knowledge a memory.

So it appears that at the knowledge level, there is nothing to say about the *structure* of these abstract bodies of knowledge called knowledge bases. This does not mean that there is nothing at all to say. Although Newell does not go into any details, he insists that "knowledge is to be characterized *functionally*, in terms of what it does, not *structurally* in terms of physical objects with particular properties and relations".

[2]The sense of 'believe' and 'know' being used here is perhaps confusing. By a 'belief', we will normally refer to something *actively* held as true. The whole point of the paper is to elucidate a sense of 'knowledge' appropriate to knowledge bases. Roughly, we intend it to refer to the logical consequences of what is believed, regardless of whether the believer is aware of the consequences or whether the beliefs are accurate.

1.2. Functionality

Newell's view of knowledge at the knowledge level is very similar to the standard notion of an abstract data type [2]: to specify what is required of a desired entity (or collection of related entities), specify the desired behaviour under a set of operations and not the structures that might be used to realize that behaviour. The canonical example is that of a *stack* that might be specified in terms of the following operations:

Create: $\rightarrow$ STACK ,
Push: STACK $\times$ INTEGER$\rightarrow$STACK ,
Pop: STACK $\rightarrow$ STACK ,
Top: STACK $\rightarrow$ INTEGER ,
Empty: STACK $\rightarrow$ BOOLEAN .

By defining these functions (over abstract stacks), we specify *what* the behaviour should be without saying *how* it should be implemented. Of course, we have not said how stacks can be used in general to do other things (like implement recursion, for instance), but we have provided the primitives of this usage. In this sense, the specification is functional.

How might similar considerations apply to knowledge? To answer this, we have to discover the primitive operations that will be composed to form complex applications of knowledge (such as problem solving, learning, decision making, and the like). At least two orthogonal operations suggest themselves immediately.

First of all, if the behaviour of an intelligent agent is to depend on what is known, a primitive operation must be to *access* this knowledge. Very roughly speaking, we have the following:

ASK: KNOWLEDGE $\times$ QUERY $\rightarrow$ ANSWER .

In other words, we can discover answers to questions using knowledge. This is the analogue of the Top, Pop, and Empty operations which were also retrieval oriented. If we consider learning to be an intelligent activity, then we need the analogue to the Push operation: we have to be able to *augment* what is known:

TELL: KNOWLEDGE $\times$ ASSERTION $\rightarrow$ KNOWLEDGE .

Again, roughly, knowledge can be acquired by assimilating new information. There are, presumably, a large number of TELL and ASK operations corresponding to the many ways knowledge can be accessed and acquired (including non-linguistic ways). There are also other kinds of operations worth considering including an analogue of Create as well as a FORGET, an

ASSUME and others. However, for our purposes, we can restrict our attention to these two operations.

1.3. Some simplifications

To be able to say anything concrete about the operations of TELL and ASK and to simplify a lot of the machinery to come, a few assumptions are necessary. First of all, if we assume that "the knowledge attributed by the observer to the agent is knowledge about the external world", then we can assume that what a knowledge base is told or asked is also about the external world. In addition, if we now restrict ourselves to *yes–no questions*, we can assume (without loss of generality) that both the *queries* and the *assertions* are drawn from the same language. The only difference between a (yes–no) *query* and an *assertion* is that a knowledge base will be *asked* if the former is true and *told* that the latter is true.

So, for some language $\mathscr{L}$ that can be used to talk about an external world, we have the following functional specification of a KB:

$$\text{TELL: } \mathrm{KB} \times \mathscr{L} \to \mathrm{KB}\,;$$
$$\text{ASK: } \mathrm{KB} \times \mathscr{L} \to \{\text{yes, no, unknown}\}\,.$$

What remains to be done is to settle on a suitable language $\mathscr{L}$ and come up with an abstract notion of a KB in terms of which these two operations can be defined.

It is perhaps worth examining at this stage how a *concrete* version of a KB could be realized. We might take $\mathscr{L}$ to be a dialect of the first-order predicate calculus and represent a KB as a sentence (or a finite set of sentences) of this language. In this case, TELL and ASK can be defined (in terms of first-order provability) by

$$\mathrm{Tell}(k, \alpha) = (k \wedge \alpha)$$

$$\mathrm{Ask}(k, \alpha) = \begin{cases} \text{yes}\,, & \text{if } \vdash (k \supset \alpha)\,; \\ \text{no}\,, & \text{if } \vdash (k \supset \neg\alpha)\,; \\ \text{unknown}\,, & \text{otherwise}\,. \end{cases}$$

There is nothing wrong with any of this, of course. This is just a standard first-order system like the kind discussed in [3] (though disguised in terms of two operations, where talk of a single query operation is much more common). However, this definition just does not address our problem any more than defining the stack operations in terms of operations on linked lists (or arrays) would. In other words, these definitions are at a different level—the symbol level—where the structure of a KB has been decided. This is not to say that there are no further design decisions to be made. In the stack case, we would

have to decide how to implement linked lists (maybe using arrays). Here, we have to decide how to realize a suitable theorem-proving behaviour over the sentences of $\mathscr{L}$ (maybe using resolution, which itself still does not say much). But the crucial choice was to *represent the knowledge symbolically* using a sentence of $\mathscr{L}$ and reduce the operations on a KB to theorem-proving operations over $\mathscr{L}$. As Newell argues,

> Logic is fundamentally a tool for analysis at the knowledge level. Logical formalisms with theorem proving can certainly be used as a representation in an intelligent agent, but it is an entirely separate issue (though one we already know much about thanks to the investigations in AI and mechanical mathematics over the last fifteen years).

Regardless of whether or not we decide to use sentences of a first-order logical language to represent what is known, a separate decision must be made about $\mathscr{L}$, the language for the TELL and ASK operations. It is the expressive power of this interface language that will determine what a KB can be said to know.[3] Since we are primarily interested in KBs whose world knowledge can be represented in first-order terms, we will start with a first-order interface language. In the next section, an appropriate first-order dialect will be defined.[4] We will then show why a more expressive *intensional* language is actually warranted. In a subsequent section, we will demonstrate that this results in abstract KB's that can still be represented in first-order terms. In sum, we will argue that a first-order language is expressively inadequate as a language for communicating with a first-order KB.

2. The Interface Language

In this section, we examine in detail some of the technical features (syntactic and semantic) a first-order dialect should have to be useful as the interface language $\mathscr{L}$. We will then show that, in fact, a language that is more powerful than this first-order language is required. This will lead to a new interface language $\mathscr{KL}$ whose use will be explained and whose semantics will be defined (by extending the semantics of $\mathscr{L}$). Finally, the TELL and ASK operations over this language will be defined.

[3]The situation is complicated here by the fact that the TELL and ASK operations may not allow precisely the same subset of the interface language to be used. As an extreme case, TELL may be restricted to atomic sentences (as in relational style databases) while ASK may allow a full first order query language. For example, you might be able to find out if $(p \vee q)$ is true but only be able to inform the KB that p is true or that q is true.

[4]The fact that we want to characterize what a KB knows at the knowledge level other than by a collection of symbolic structures will force us to use the *semantics* of $\mathscr{L}$ directly in our definitions. While this may present a problem for those readers more familiar with symbolic or *proof-theoretic* uses of formal languages, we have, at least, designed the semantic theory of $\mathscr{L}$ with the specific goal of TELL and ASK in mind.

2.1. A first-order language

As pointed out in [4], it is one thing to say that *some* first-order language is going to be used for some purpose but quite another matter to say *which*. A traditional use of logical languages in AI has been to *represent* knowledge symbolically and has lead to considerations such as prenex forms, Skolemization and various theorem-proving strategies. Our demands are quite different in that we seek a logical language $\mathcal{L}$ as a medium for querying and updating a KB.

2.1.1. *Singular terms*

The first and perhaps most radical characteristic of the language $\mathcal{L}$ is its treatment of *singular terms*. While there is a sense in which constant and function symbols are theoretically dispensable in first-order settings (see, for example, [5, pp. 84–85]), in our case, some form of singular term is necessary. Specifically, if we ever want to be able to ask a KB a *wh-question* and obtain an answer other than *unknown*, we must be able to provide the information necessary to answer the question. We have to not only be able to tell the KB that there exists *someone* with a certain property; we must be able to tell it *who* that individual is and have the KB realize that it is being told precisely that.

For example, if we consider telling a KB that[5]

$$\text{CoastalCity}(\text{LargestCity}(\text{USA}))\,,$$

we must be clear as to whether we are trying to tell it *what city* is a coastal city or merely (and more plausibly) *that some city* (that also happens to be the largest American city) is. But under what condition do we want to say that the KB knows what the largest American city is? It certainly does not amount to simply knowing something like

$$(\text{LargestCity}(\text{USA}) = \text{Birthplace}(\text{EdwardMcDowell}))$$

unless the KB also knows where McDowell was born. But then we might argue that even if the KB knows that

$$(\text{LargestCity}(\text{USA}) = \text{NewYorkCity})\,,$$

it still does not know the largest city unless it knows what city New York City is. To allow a KB to answer wh-questions, we have to *design* $\mathcal{L}$ in such a way as to avoid this potentially infinite regress.

[5]Here and elsewhere, the reverse PROLOG convention will be used: variables will appear in lower case, while function, constant and predicate symbols will begin in upper case.

One way to do this is to consider the entire space of singular terms as being partitioned into equivalence classes where, intuitively, two terms are in the same class if they corefer. Then we might say that a KB knows what the largest city is if it knows what equivalence class the term belongs to. To inform the KB of this, we need only notice that since there are countably many singular terms in $\mathscr{L}$, there can be at most countably many equivalence classes. So we can simply introduce into $\mathscr{L}$ a countable number of new terms (which we call *parameters*)

$$1, 2, 3, \ldots$$

understood as distinct but semantically vacuous representatives of each equivalence class.[6] By convention, then, to know what equivalence class a term t belongs to (and therefore be able to answer the wh-question) is to know something of the form

$$(t = k) \quad \text{for some parameter } k\,.$$

So, for example, if a KB knows that

$$(\text{NewYorkCity} = \text{TheBigApple}) \wedge (\text{TheBigApple} = 2)\,,$$

then we will say it knows what city New York and The Big Apple are. Of course, there is at least one sense in which this is contentious or just plain false—knowing what city New York is *should* involve New York somehow. But recall that the sense of 'know' we have in mind is a conventional one where first, it makes sense for a KB to have knowledge at all and second, this depends on what questions it can answer, not on what connections it has to the world. The issue here is not how well acquainted a KB is with New York City, but rather when a certain wh-question can be answered.

2.1.2. *Incomplete knowledge*

Given the above discussion of wh-questions, the only other constraint that plays a role in determining the form of $\mathscr{L}$ is the *incompleteness* of knowledge bases. As argued in [6, 7], to allow a knowledge-based system to capitalize on whatever knowledge about its application domain is available, it must be possible for the KB to find out about the world in an incremental way. The only alternative is to postpone the assimilation of information until it has become sufficiently specific which, for many applications, may simply *never*

[6]Parameters are thus *standard names* of members of the equivalence classes. Alternately, we could introduce some special way of stating for each number i and term t that t belongs to the ith equivalence class.

happen. In terms of TELL and ASK, what this amounts to is that $\mathcal{L}$ has to be sufficiently expressive to allow us to make weak assertions about the world.

Consider, for example, the sentence[7]

$$\text{LosAngeles} \neq \text{Capital(California)}.$$

The question is why (apart from convenience) we would ever need a sentence like this in $\mathcal{L}$. Why not instead just tell the KB what the capital of California is:

$$\text{Capital(California)} = \text{Sacramento}$$

(assuming it already knows that Sacramento and Los Angeles are distinct cities)? From this sentence, it is a simple matter to infer the negated sentence. Similarly, why would we ever bother telling the KB that

$$(\text{Capital(California)} = \text{SanFrancisco}) \vee (\text{Capital(California)} = \text{Sacramento})$$

since the disjunction can be easily inferred once we tell it what the capital is? The point should be clear: in terms of communicating with a KB, it's not so much that negation and disjunction allow information to be conveyed that otherwise could not, but only that they do so *without also requiring more information* than what may be available (such as knowing what the capital is). In other words, they allow a KB to be told *exactly* what is known about the world, however vague. If more specific information ever becomes available, then it will be assimilated as well, but until that point, the KB can at least use whatever incomplete knowledge it has acquired.

Much the same argument can be made for the inclusion in $\mathcal{L}$ of existential quantification and non-parameter terms (which need not corefer). The conclusion to be drawn is that the standard features of a first-order language such as non-parameter terms, negation, disjunction and existential quantification, can all be motivated from the point of view of incomplete knowledge bases; they all allow knowledge to be acquired at the appropriate level of vagueness or generality.[8] If the source of information was suitably rich, the expressive power would not be necessary (though obviously convenient, in the case of universal quantification).

[7]The term 'sentence' is being used technically here to mean a closed formula of $\mathcal{L}$.

[8]Interestingly enough, second- (or ω) order quantification (over properties, relations and the like) is much more difficult to motivate in these terms. While it makes sense to want to be able to say that every major city of New York has a certain property (without having to know what cities are the major ones) it seems strange to want to say anything about *every property* of New York (including the negation of every property it does not have).

2.1.3. *Semantics*

The language $\mathscr{L}$ we are considering is built up in the obvious recursive way from a set of function and predicate symbols of every arity (including constants, the 0-ary function symbols), parameters (which behave syntactically exactly like constants), (individual) variables, and the usual logical punctuation. The function symbols (including the constants) and the predicate symbols (excluding the equality symbol) are considered to be the *non-logical* symbols; all others are *logical* symbols. Expressions in the language are syntactically divided into *terms* and *sentences*: the terms are variables, parameters and function applications; the sentences are predicate applications, equality expressions, negations, disjunctions and quantifications. Sometimes disjunction will be used as *the* binary propositional connective and, other times, material implication. Similarly, existential or universal quantification will be *the* quantifier. The intent is that any other connective (such as material equivalence) be made available in the usual way. For any variable x, closed term t and expression α (term or sentence), the expression α_t^x is the result of replacing all free x in α by t. Finally, if $x_1, x_2, \ldots, x_n$ are the free variables in α, then $\alpha[t_1, t_2, \ldots, t_n]$ is the closed expression resulting from simultaneously replacing each free x_i by t_i.

We will describe the semantics of this language using what Quine has called *truth value semantics*.[9] While this form of semantic specification does not shed any light on the referential nature of the language, it forms a more convenient semantic basis for the language $\mathscr{KL}$, to be introduced later. Essentially, the idea is to specify (with as little fuss as possible) which of all the possible mappings from sentences of $\mathscr{L}$ to true or false are admissible interpretations.

Consider, for example, an assignment of truth values to sentences of the form $t_1 = t_2$. The kind of thing we obviously want to rule out is an assignment where, say, the three sentences

$$f(1) = 2, \qquad c = 1 \quad \text{and} \quad f(c) = 3$$

are all interpreted as true.

The easiest way to do this is to make the assignment to equality sentences depend on a partitioning of the primitive terms of the language. The *primitive terms* are those that contain exactly one function symbol. So, for example,

$$c, \qquad f(1) \quad \text{and} \quad g(2, 1)$$

are all primitive terms; if these all belong to the same equivalence class as the parameter 1, then so must the non-primitive terms

$$f(c), \qquad f(g(2, c)) \quad \text{and} \quad g(2, f(f(1))).$$

[9]See [8], for an introduction to (and a defense of) this kind of semantic theory.

To enforce this constraint, we first assign a parameter (or equivalence class) to each primitive term. Given v, an assignment of this kind, we can define what it means for any two terms of $\mathscr{L}$ to corefer as follows:

The *coreference relation* (given v) is the least set of pairs such that:

(1) if t is a primitive term, then t and $v[\![t]\!]$ corefer;

(2) if t_1 and t_2 corefer, then so do $t^x_{t_1}$ and $t^x_{t_2}$.

It is a simple matter to show that for each v, this assigns a unique coreferring parameter (and thus, a unique equivalence class) to each term of the language. In particular, this definition of coreference guarantees that *1* and *2* do not corefer on any v.

An assignment of truth values to sentences of $\mathscr{L}$ is much like an assignment to terms of $\mathscr{L}$ except that there are only two possible equivalence classes: the true sentences and the false ones. As above, the easiest way to rule out inadmissible interpretations is to make them depend on assignments to the primitive sentences (and for equality sentences, to the primitive terms). The *primitive sentences* are the atomic ones that contain no function symbols (so all primitive expressions, terms or sentences, have exactly one non-logical symbol). Given a set s of primitive sentences (taken to be the true ones) and an assignment v as above (taken to specify coreferentiality), the truth value of every sentence of $\mathscr{L}$ can be defined as follows:

The *true sentences* on s and v form the least set such that

(1) every element of s is true on s and v;

(2) if t_1 and t_2 corefer given v, then $(t_1 = t_2)$ is true on s and v and the atomic sentences $\rho^x_{t_1}$ and $\rho^x_{t_2}$ have the same truth value on s and v;

(3) if α is not true on s and v, then $\neg\alpha$ is;

(4) if either α or β is true on s and v, then so is $(\alpha \vee \beta)$;

(5) if for some parameter i the sentence α^x_i is true on s and v, then so is $\exists x\alpha$.

The definition can be extended in the usual way to include conjunctions, implications, equivalences and universal quantifications based on negations, disjunctions and existential quantifications. Given this definition of truth, we define a sentence as *valid* if it is true on every s and v. A set of sentences is *satisfiable* if there is an s and v on which every element of the set is true.

The first thing to notice about the above definition is that when we restrict our attention to that subset of $\mathscr{L}$ without parameters or equality, it produces the same valid and satisfiable sentences as the standard one due to Tarski. As for equality, to the extent that it is considered to be part of a first-order language at all, it is usually treated as a regular predicate with a set of axioms in a theory forcing the predicate to be an equivalence relation. Semantically, as far as the language is concerned, this leaves the predicate symbol uninterpreted. In our case, however, the language is constrained to have a very definite interpretation of equality. Specifically, once an assignment of equivalence classes to terms has been made, the interpretation of equality sentences is fixed.

One major difference between our definition of validity and a more standard one is that in our case only, the following sentences are all valid:

$$\begin{aligned}
&\neg\exists x_1\,\forall y(y = x_1)\,,\\
&\neg\exists x_1\,\exists x_2\,\forall y(y = x_1) \vee (y = x_2)\,,\\
&\neg\exists x_1\,\exists x_2\,\exists x_3\,\forall y(y = x_1) \vee (y = x_2) \vee (y = x_3)\,,\\
&\qquad\vdots
\end{aligned}$$

In Tarskian terms, what this amounts to is a restriction on interpretations forcing the domain of quantification to be infinite. In practice, this is not a very serious restriction since in no way does it constrain the interpretation of the predicate (and function) symbols, all of which may be taken to have finite or infinite extensions. It is only sentences that talk about the totality of what exists that will have different interpretations. These sentences are indeed special in our case since they contain only logical symbols and so are either valid or unsatisfiable. Another way of looking at this is to say that there is no real way in $\mathscr{L}$ to talk about what exists as a whole except by talking about the extensions of the predicate or function symbols. This suggests that a *typed* form of quantification might be more compatible with this semantics. This would syntactically rule out the above list of valid sentences which talk about the size of the entire domain.

One final point to observe about the semantics of $\mathscr{L}$ is that the Compactness Theorem fails for this dialect. In other words, just because every finite subset of a set of sentences is satisfiable does not mean that the set itself is. An example of such a set is $\{\exists x\phi(x), \neg\phi(1), \neg\phi(2), \neg\phi(3), \ldots\}$ for some monadic predicate ϕ. Again this has little practical consequence since our major concern will be with sentences and finite sets of sentences of $\mathscr{L}$.

So, to summarize, the truth theory of $\mathscr{L}$ is based on a partition of the primitive sentences into two groups and a partition of the primitive terms into an infinite number of groups. Together, the partitions determine the truth value of every sentence in the language. Thinking of the language $\mathscr{L}$ as talking about some (possible) world (or world state, situation, state of affairs, slice of reality), an s and a v together tell us exactly what that world is like in as much detail as the language $\mathscr{L}$ allows. We will consequently call a pair $\langle s, v\rangle$ a *world structure* since it specifies a world relative to the language $\mathscr{L}$. Of course, many worlds can correspond to such a structure, namely those whose differences are not captured by the language $\mathscr{L}$.

2.1.4. *Proof theory*

The semantics for $\mathscr{L}$ presented above does not require dealing with formulas having free variables. The proof theory will also have this property in that at

any stage of a proof, only *sentences* of $\mathscr{L}$ will be considered, with parameters playing the role of free variables.

The major deviation from standard axiomatizations of first-order logic involves the relationship between quantification and parameters. Specifically, to guarantee soundness and completeness (with the above semantics), we must insure that

$$\forall x\alpha \text{ is a theorem} \quad \text{iff} \quad \text{for every parameter } k,\ \alpha_k^x \text{ is a theorem}.$$

The 'only if' part of this is easy to guarantee with an axiom schema (usually called the Axiom of Specialization)

$$\forall x\alpha \supset \alpha_t^x \quad \text{for any term } t.$$

The modification to the rule of Universal Generalization necessary to handle the converse is a bit more troublesome. The obvious 'rule' is

$$\text{From } \alpha_1^x, \alpha_2^x, \ldots, \alpha_i^x, \ldots \text{ infer } \forall x\alpha$$

which unfortunately leads to a notion of proof that requires an *infinite* number of subproofs before a universal can be concluded. On the other hand, one might be tempted to reason as follows:

> Suppose I am able to prove a theorem α which contains some parameter k. There is nothing special about k, so I should be able to also prove the theorem for any other parameter. Consequently, I am justified in concluding that $\forall x\alpha'$ where α' is α with k replaced by x.

Except for the equality predicate, this reasoning is sound. The trouble with equality is that the parameter k is indeed special in that it is the only parameter that is equal to k and to no others. In particular, we do not want to be able to reason

$$\ldots 1 \neq 2, \text{ and therefore, } \forall x(x \neq 2) \ldots$$

since *1* is special in not being equal to *2*. So the reasoning in general has to be as follows:

> Suppose I am able to prove α which contains parameter k as well as parameters $i_1, \ldots, i_n$. If I am also able to prove α but with k replaced by $i_1, \ldots, i_n$, then there is no way the theorem could have been a consequence of the fact k is special relative to the other i_j. Thus, I can conclude that $\forall x\alpha'$.

As it turns out, *this* reasoning is sound[10] and leads to the following version of Universal Generalization:

From $\alpha^x_{i_1}, \ldots, \alpha^x_{i_n}$, infer $\forall x\alpha$ provided the i_j range over all the parameters in α and at least one not in α.

The only other aspect of the proof theory of $\mathscr{L}$ to worry about is the treatment of equality itself. Fortunately, the parameters permit a very simple and elegant formalization. All that is required is a single axiom schema (called the Axiom of Equality) to state that only identical parameters are equal:

$(i = i) \wedge (i \neq j)$ for any two distinct parameters i and j.

The fact that this axiom schema has a proviso attached is of no real consequence. Standard formulations of first-order logic also attach a proviso to the Axiom of Specialization to avoid the collision of quantifiers. In [7], it is shown that all the usual properties of equality (like the Leibniz property) can be derived from this axiom. For example, we can prove that equality is a symmetric relation by Universal Generalization from

$$\forall y(1 = y) \supset (y = 1),$$

which derives from

$$(1 = 1) \supset (1 = 1) \quad \text{and} \quad (1 = 2) \supset (2 = 1),$$

both direct consequences of the Axiom of Equality. So, to summarize, we have the following proof theory for $\mathscr{L}$:

Axioms.

(A1) $\alpha \supset (\beta \supset \alpha)$.
(A2) $(\alpha \supset (\beta \supset \gamma)) \supset ((\alpha \supset \beta) \supset (\alpha \supset \gamma))$.
(A3) $(\neg\beta \supset \neg\alpha)) \supset ((\neg\beta \supset \alpha) \supset \beta)$.
(AD) $\forall x(\alpha \supset \beta) \supset (\forall x\alpha \supset \forall x\beta)$.
(AS) $\forall x\alpha \supset \alpha^x_t$.
(AE) $(i = i) \wedge (i \neq j)$ *for all distinct* i, j.

[10]See [7] for details. Note also that the reasoning remains sound when 'provable' is replaced above by 'valid'. However, it is unsound when 'provable' is replaced by 'true'. To conclude that $\forall x\alpha$ is true, we would need *all* instances of α^x_k as evidence.

Rules of inference.

(MP) *From* α *and* $\alpha \supset \beta$, *infer* β.

(UG) *From* $\alpha^x_{i_1}, \ldots, \alpha^x_{i_n}$ *where the* i_j *are those in* α *and one not in* α, *infer* $\forall x \alpha$.

This gives us a precise notion of theoremhood in $\mathscr{L}$ that is extensionally equivalent to the above definition of validity. That is, it can be shown that

$$\vdash \alpha \quad \text{iff} \quad \alpha \text{ is valid}.$$

So it should be no surprise that the theorems (at least those without parameters and equality) correspond to those of the standard axiomatization.

2.2. An extended interface language

The above definition of validity and theoremhood can be used to define the two interface functions TELL and ASK that would use $\mathscr{L}$ to link a KB to an external agent (or program) that is using the KB as a repository for information about some application domain. However, we will argue that the language $\mathscr{L}$ is an appropriate query language for ASK only if the language for TELL is limited in its use of negation, disjunction and existential quantification. The expressive power of $\mathscr{L}$ as an assertion language simply makes it too weak as a query language.

2.2.1. *First-order limitations*

Consider, for example, a KB that is keeping track of state capitals:

Capital(Pennsylvania) = Harrisburg
Capital(Indiana) = Indianapolis
.
.
.

Conceptually, at least, there is certainly no problem with a wh-question

Capital(state) = city?

which asks the KB to list the states and their capitals (by using free variables *state* and *city*). This is in fact what would happen with a PROLOG data base and query similar to the above. The trouble arises when we consider a KB that has much less knowledge about this domain. For example, a KB may only know

Capital(Texas) = Houston $\vee$ Capital(Texas) = Austin
Capital(California) $\neq$ LosAngeles

without having identified the capital of either state.[11] In this case, the request to list the states and their capitals is much more problematic. In general (and at best), the system will be able to provide some *description* of the capitals based on what it knows such as "The capital of California is not a coastal city". But equally *true* would be to say "The capital of California is a city that is not located in Wyoming nor in New York". To avoid *that* kind of answer, some measure of relevance would have to be established based, presumably, on a theory of what an acceptable answer in this domain should be and what the user already knows. One can imagine, for example, a system engaging in a clarification dialogue with the user about an acceptable form of answer. In other words, this kind of ability for wh-questions goes beyond a simple interface to a KB and starts looking more and more like a separate knowledge-based system with a KB of its own. There is, in fact, a minor problem even in the first case if terms like 'Harrisburg' in the above are really *constants*, not parameters. For example, a KB could contain $\phi(c)$ for a large number of constants c while still knowing that there was a unique ϕ. Once again, it is not clear what the answer to a wh-question should be.

To the extent that a KB is intended to provide a service to a knowledge-based system, it should, at the very least, be able to inform a user about when a wh-question is problematic. From there, it would be a separate step to decide what to do about an answer. For example, consider the question that asks for the capital of California:

$$\text{Capital(California)} = \text{city} .$$

This question has a direct answer only when the system knows what the capital is. To find out if the KB has that knowledge, the question

$$\exists \text{city}[\text{Capital(California)} = \text{city}]$$

is not adequate since this asks if California has a capital and not if the system knows what it is. In other words, the kind of question we have to ask is not whether or not there is something that *is* the capital but whether or not there is something that *the* KB *believes to be* the capital. The difference is crucial: the first question asks the KB about the *world*, whether or not California has a capital; the second question asks the KB about the KB *itself*, whether or not it has on record the capital of California.

If we accept the position that the language $\mathscr{L}$ is intended to talk about the world only, then we have to go beyond $\mathscr{L}$ to be able to ask questions that also

[11]This is, of course, precisely the kind of knowledge that *cannot* be represented in the subset of first-order logic embodied in PROLOG. Whatever its merit as a programming formalism, PROLOG is severely limited as a representation language.

talk about the KB. The interface language $\mathscr{KL}$ contains all of $\mathscr{L}$ and, in addition, for any formula α contains a formula $\boldsymbol{K}\alpha$. A sentence $\boldsymbol{K}\alpha$ here should be read as "The KB currently knows that α". Among the differences between the two sentences, note that if the question α (i.e., is α true?) is answered *unknown*, then the question $\boldsymbol{K}\alpha$ (i.e., is α known?) should be answered *no*. Before describing the exact semantics of the language $\mathscr{KL}$, it perhaps worth examining how it can be used as an interface language.

2.2.2. *The use of* $\mathscr{KL}$

The most important property of $\mathscr{KL}$ is that whereas we can use a question like

$$\exists \text{city}[\text{Capital}(\text{California}) = \text{city}]$$

to find out if California has a capital, we can now also use

$$\exists \text{city}\boldsymbol{K}[\text{Capital}(\text{California}) = \text{city}]$$

to find out if the KB knows any of them. In other words, the KB will answer *yes* to the first one if it thinks California has a capital and to the second one if it thinks it knows one of them. The first sentence is true of a *world* where California has a capital; the second is true of a *knowledge base* that knows a capital of California. Similarly, the sentence

$$\exists \text{state} \neg \exists \text{city}\boldsymbol{K}[\text{Capital}(\text{state}) = \text{city}]$$

is true of a KB when there is a state such that no city is known by the KB to be its capital.

There are also sentences in $\mathscr{KL}$ that talk simultaneously about the world and the KB. The sentence

$$\exists \text{city}[\text{MajorCity}(\text{city}, \text{California}) \wedge \neg \boldsymbol{K}\text{MajorCity}(\text{city}, \text{California})]$$

will be true if California really has a major city that the KB does not know to be a major city. For this sentence to be true, the world and the KB have to be in a certain state. If asked as a question, we are asking the KB for an opinion on the relationship between what is true and what it knows. The answer can be *yes*, *no* or even *unknown*. For example, if the KB knows that

$$\forall \text{city}[\text{MajorCity}(\text{city}, \text{California}) \equiv (\text{city} = \text{LosAngeles})],$$

then, assuming the KB knows what city LosAngeles is, the answer would be *no* since the KB knows there can be no major city of California that it does not

know about. Of course, the KB can be mistaken about California and major cities, but that is a separate issue. Given the above belief, the KB is certainly justified in believing that it knows all the major cities of California. On the other hand, if the KB only knows that

$$\text{MajorCity(Fresno, California)} \vee \text{MajorCity(Livermore, California)},$$

then the answer to the question is *yes* since the KB knows that California has a major city that is not among the known ones. Although it does not know *what* city has that property, it does know that there is one, and moreover, it is either Fresno or Livermore. The reasoning here, from the point of view of the KB, is somewhat subtle. The KB knows that neither Fresno nor Livermore are known to be major cities. Moreover, it knows that one of them is a major city. Thus, it knows that there is a major city that is not known to be a major city. Finally, in the simplest case where the KB has no information at all about the major cities, the answer is *unknown* since the KB has no way of knowing whether or not California has major cities it might not know about.

The above sentence can be generalized in an interesting way. We can first abstract from California in the above and get

$$\exists \text{state} \neg[\exists \text{city}(\text{MajorCity}(\text{city}, \text{state}) \wedge \neg \boldsymbol{K}\text{MajorCity}(\text{city}, \text{state}))].$$

This sentence will be true if, for some state, the KB knows all of its major cities. On the other hand, the truth of the sentence

$$\exists \text{state} \boldsymbol{K} \neg[\exists \text{city}(\text{Majorcity}(\text{city}, \text{state}) \wedge \neg \boldsymbol{K}\text{MajorCity}(\text{city}, \text{state}))]$$

once again depends only on the KB and not on the world. Roughly speaking, this sentence will be true if the KB knows a state for which it thinks it knows all the major cities.[12] This sentence talks about the *meta-knowledge* of a KB, that is, the knowledge it has about what it knows about the world. This can be seen in the syntax of the sentence itself where a $\boldsymbol{K}$-operator occurs within the scope of another $\boldsymbol{K}$-operator.

So far, we have concentrated on how a language like $\mathcal{KL}$ can be used as a query language to ask the KB not only about what it thinks is *true*, but what it thinks is *known*. But $\mathcal{KL}$ can also be used to *tell* a KB about the world by referring to what is already known. To see this, we must first examine how $\mathcal{KL}$ *cannot* be used as an assertion language.

Consider, for example, a sentence of $\mathcal{KL}$ of the form $(\boldsymbol{K}\alpha \vee \boldsymbol{K}\neg\alpha)$. Intuitively, this sentence is true of a KB when that KB either knows α or knows

[12]More precisely, the sentence is true if there is something (say, a state) that has the property that the KB thinks that there is no city that is a major city of that state without the KB knowing it.

$\neg\alpha$. As a question to a KB, the answer to this sentence should be *yes* if the KB knows whether α is true and *no* otherwise. Suppose we now consider telling a KB that this sentence is true. If the KB already knows the truth value of α, the sentence does not tell the KB anything about itself that it does not already know and so the assertion is redundant. If, on the other hand, the KB does not know the truth value of α, then telling it the sentence is certainly not going to change anything since it will not help the KB decide what the truth value is. In fact, the KB already believes the sentence to be false and the only way it can (and should) change its mind is when it discovers the truth value of α. In this case, then, the assertion of the sentence contradicts what is already known. A KB cannot be informed of the truth value of α simply by being told that it knows the truth value. Such an assertion will either be redundant or contradictory.

The problem with asserting $(\boldsymbol{K}\alpha \vee \boldsymbol{K}\neg\alpha)$ is that this sentence does not say anything at all about the world. It is true or false based only on the current state of the KB. Its truth, therefore, cannot possibly affect the picture of the world that the KB might have. But there are sentences of $\mathscr{KL}$ (over and above those of $\mathscr{L}$) that have this property. For example, the sentence

$$\forall \text{city}[\text{MajorCity}(\text{city}, \text{Wyoming}) \supset \boldsymbol{K}\text{MajorCity}(\text{city}, \text{Wyoming})]$$

is true when the KB knows every major city of Wyoming. As a question, the KB will answer *yes* if it thinks it knows every major city, *no* if it thinks it does not and *unknown* otherwise. An instance of this last case is when all the KB knows is that

$$\text{MajorCity}(\text{Cheyenne}, \text{Wyoming}).$$

In this case, it *does* make sense to inform the KB that it knows all the major cities since, in a roundabout way, this tells it something about the world, namely that Cheyenne is the *only* major city of Wyoming. As discussed in [6, 9], this is an instance of the *closed world assumption*, relativized to the major cities of Wyoming. The sentence in fact tells the KB that any city not *known to be* a major city of Wyoming *is not* a major city of Wyoming. Similarly, telling a KB that

$$\forall \text{city}[\exists \text{state}\ \text{MajorCity}(\text{city}, \text{state}) \equiv \exists \text{number}\boldsymbol{K}(\text{Population}(\text{city}) = \text{number})]$$

tells it that the major cities are exactly those cities whose populations are (currently) known. Finally, telling a KB that

$$\exists \text{city}[\text{MajorCity}(\text{city}, \text{NewYork}) \wedge \neg\boldsymbol{K}\text{MajorCity}(\text{city}, \text{NewYork})]$$

is a way of asserting something about New York, that is, that it has a major city that the KB does not know about.

Given a more general interface language like $\mathscr{KL}$, the definition of TELL and ASK must be reconsidered. Specifically, we can no longer use the intuition that a question α must be answered *yes* when it is a (first-order) consequence of what is in the KB, since α now may contain $\boldsymbol{K}$ operators. However, we do know that α should be answered *yes* when it is known to be true, that is, precisely when $\boldsymbol{K}\alpha$ is true. So the answer to questions in this more general framework is determined by the truth conditions of the language $\mathscr{KL}$, which we will now examine.

2.2.3. *Competence and closure*

The semantics of $\mathscr{KL}$ is based on two assumptions about the sense of knowledge we are considering. The first assumption, which was motivated above, is the Assumption of Competence. The second assumption, which deals with meta-knowledge only, is called the Assumption of Closure. In this section, we examine these two assumptions in preparation for a truth theory of $\mathscr{KL}$ based on the one for $\mathscr{L}$.

The first assumption might be phrased as follows:

Assumption of Competence. *Every logical consequence of what is known is also known.*

Again this is not to say that a KB actively believing α and $(\alpha \supset \beta)$ must somehow be *logically forced* into actively believing β. Rather (in the case of sentences of $\mathscr{L}$ anyway), it says that if a KB has a picture of a world where both α and $(\alpha \supset \beta)$ are true, then the world it imagines must be one satisfying β.[13] This assumption places a lower limit on what knowledge a KB can have since it forces every KB to know at least the valid sentences and the consequences of what it knows.

An upper limit on what a KB can know is also determined by this assumption. If a KB knows both α and $-\alpha$, then *every* sentence of $\mathscr{KL}$ must be known. In other words, any KB that has inconsistent knowledge must know exactly the same sentences. It is important to realize that there is absolutely nothing contradictory about a KB that is inconsistent. Nor is it a defect of logic that an inconsistent KB knows every sentence. It is just a property of our definition of negation in $\mathscr{L}$ that a sentence and its negation cannot be simultaneously satisfied. What is implicit in a picture of a world where both a

[13]This is simply a consequence of the truth conditions of the language. There is no notion of the KB *doing* anything here (like a proof, for instance), but only us examining the consequences of what it knows.

sentence (of $\mathscr{L}$) and its negation are true? It is simply the case that in every world so described *of which there can be none*, any sentence of $\mathscr{L}$ is true. As it turns out, however, the Assumption of Closure described below will constrain a KB to be *consistent*, so the point is really moot.

This is the only assumption about world knowledge that will be made. Specifically, there is no assumption that the world knowledge of a KB is *complete*. In other words, there may be sentences of $\mathscr{L}$ that are neither known to be true nor known to be false. In addition, there is no assumption that the world knowledge is *accurate*. Just because the KB thinks that something is true does not mean that it *really* is. While ideally it would be nice to have a KB that was a complete and accurate model of reality, the well-formedness of a KB does not depend on these two properties.

One way of looking at the competence of a KB semantically is to think of a KB as an incomplete picture of some world, a world where what the KB knows is true. In general, there will be many distinct worlds that satisfy everything that the KB knows.[14] We can call these worlds *compatible* with the KB. For example, if a KB knows α and $(\sigma \vee \beta)$, then a world satisfying $\alpha, \neg\beta$, and σ is compatible with the KB but one satisfying $\neg\alpha$ is not. Given this characterization of compatible worlds, the Assumption of Competence tells us that a sentence of $\mathscr{L}$ is known exactly when it is true in all compatible worlds. So we can identify in an abstract way the *world knowledge* held by a KB with its set of compatible worlds. In other words, if we want to treat the knowledge held by a KB as a distinct abstract entity, a set of worlds seems to be a reasonable abstraction to use. We can then use a *set* of world structures to specify a KB as anything having this world knowledge.

This set theoretic view of knowledge bases gives us a natural picture of knowledge acquisition. One can imagine starting with a KB that has no world knowledge and so is described by the set of *all* world structures. If it now finds out that α is true, any world where α is false is no longer compatible with the KB and so the KB is described by a smaller set of world structures. As more and more knowledge is acquired, more and more worlds are eliminated. In the limit, one can imagine a KB knowing the truth value of every sentence of $\mathscr{L}$ and thus having narrowed its picture of the world down to the point where only the worlds corresponding to a *single* world structure satisfy what it knows. This is as far as we can go using $\mathscr{L}$. Any additional non-redundant information leads to an inconsistent KB where the set of compatible worlds is empty.

If we are assuming that a KB can address any question phrased in $\mathscr{KL}$, then we are considering a KB with meta-knowledge as well as world knowledge. In fact, implicit in the above discussion was that a KB could answer questions about its own knowledge accurately and even completely. If we call a sentence

[14]This might be true even if the KB is *complete*, that is, knows the truth value of every sentence of $\mathscr{L}$, since there may be aspects of worlds that $\mathscr{L}$ does not address.

of $\mathcal{KL}$ *pure* if it is just about the knowledge of the KB (that is, is true or false independently of the world),[15] then we have the final assumption:

Assumption of Closure. *A pure sentence is true exactly when it is known.*

This assumption states that no matter how incomplete or inaccurate the *world* knowledge of a KB may be, it always has complete and accurate knowledge about its own knowledge. Of course, the assumption has to be understood in the context of the competence assumption. The KB need not actually derive all true sentences about itself. The knowledge is only *implicitly* there. On the other hand (and as discussed above), the assumption is that there will never be any reason to inform the KB about itself; we can concentrate on providing it world knowledge and leave it the responsability of keeping its meta-knowledge up to date. But more importantly, there is never any reason to doubt what a KB knows about itself; it is the final authority on what it does and does not know.

The last property is enough to guarantee the *consistency* of a KB. The reason is that an inconsistent KB would have to know every sentence including one saying that it did not know another. But, by the closure assumption, it would have to be true that it did not know the latter sentence, contradicting the assumption that it knows every sentence.

The two assumptions together tell us how to interpret any sentence $\boldsymbol{K}\alpha$ given the knowledge of a KB characterized as a set of worlds. If α is pure, then $\boldsymbol{K}\alpha$ will be true whenever α is. If α is a sentence of $\mathcal{L}$, $\boldsymbol{K}\alpha$ will be true iff every compatible world satisfies α. Finally the case where α is neither pure nor first-order will be handled by appealing to the first two cases.

2.2.4. *The semantics of* $\mathcal{KL}$

Given this informal interpretation of the $\boldsymbol{K}$ operator, we can present a truth value semantics for $\mathcal{KL}$ similar to the one presented earlier for $\mathcal{L}$. Since $\mathcal{KL}$ includes all of $\mathcal{L}$ as a special case, an interpretation of the sentences of $\mathcal{KL}$ depends on both a world and a KB. Thus we require a specification of both a world and a KB to determine the truth value of every sentence. As before, we can take a world structure to be a pair $\langle s, v\rangle$ where s is a set of primitive sentences and v is an assignment of parameters to primitive terms. A KB *structure* can be taken to be any non-empty set of world structures, intuitively, those that are compatible with the KB.[16] If k is a KB structure, then given a world structure consisting of s and v, we can define the truth value of any sentence of $\mathcal{KL}$ as follows:

[15]Syntactically, a sentence of $\mathcal{KL}$ is *pure* iff every predicate symbol (excluding equality) and every function symbol (including constants) appears within the scope of a $\boldsymbol{K}$.

[16]As we will discuss later, not every set of world structures is appropriate here.

The *true sentences* on s, v and k form the least set such that:

(1) every element of s is true on s, v and k;

(2) if t_1 and t_2 corefer given v, then $(t_1 = t_2)$ is true on s, v and k and the atomic sentences $\rho^x_{t_1}$ and $\rho^x_{t_2}$ have the same truth value on s, v and k;

(3) if α is not true on s, v and k, then $\neg\alpha$ is;

(4) if either α or β is true on s, v and k, then so is $(\alpha \vee \beta)$;

(5) if for some parameter i the sentence α^x_i is true on s, v and k, then so is $\exists x\alpha$.

(6) if α is true on s', v' and k for every $\langle s', v'\rangle$ in k, then $\boldsymbol{K}\alpha$ is true on s, v and k.

This definition duplicates the one for $\mathscr{L}$ except for the last clause which stipulates that $\boldsymbol{K}\alpha$ is true on k iff α is true on every element of k (and k itself, if α is not first-order). If α is first-order, then a KB structure k knows α just in case α is true (in the first-order sense defined earlier) on every pair $\langle s, v\rangle$ that is an element of k. As with $\mathscr{L}$, we define a sentence to be *valid* if it is true on every s, v and k. A set of sentences is *satisfiable* if there is an s, v and a k on which every element of the set is true.

Since our main objective is to use $\mathscr{KL}$ as an interface language, it is the semantics of $\mathscr{KL}$ and how it leads to a definition of TELL and ASK that concerns us. If we were interested in formulating theories and reasoning about KB's, then the proof theory of $\mathscr{KL}$ would be useful to study. In fact, we are not so much interested in *describing* and reasoning about a KB here, but in actually *specifying* one and using it to reason about the world. For the sake of completeness, however, we present an axiomatization of $\mathscr{KL}$ that has the same rules of inference as $\mathscr{L}$ and the following axioms.[17]

Axioms. *Those of* $\mathscr{L}$ *(with modification to* AS*) and*

(KAX) $\boldsymbol{K}\alpha$ *where* α *is an axiom of* $\mathscr{L}$.

(KMP) $(\boldsymbol{K}\alpha \wedge \boldsymbol{K}(\alpha \supset \beta)) \supset \boldsymbol{K}\beta$.

(KUG) $\forall x\boldsymbol{K}\alpha \supset \boldsymbol{K}\forall x\alpha$.

(KCL) $\alpha \equiv \boldsymbol{K}\alpha$ *provided* α *is pure*.

The reason the Axiom of Specialization must be modified has to do with its interaction with the $\boldsymbol{K}$-operator. Consider, for example, a KB that thinks it knows all the major cities of Ohio. For this KB, the sentence

$$\forall \text{city}[\boldsymbol{K}\text{MajorCity(city, Ohio)} \vee \boldsymbol{K}\neg\text{MajorCity(city, Ohio)}]$$

is true, in that for each city, the KB either believes that it is or believes that it is not a major city of Ohio. Indeed, this sentence follows from

[17]In [7] there is a Henkin style proof that this axiomatization is both sound and complete with respect to the semantics given above.

$\boldsymbol{K}\forall x$[MajorCity(x, Ohio) $\supset$ $\boldsymbol{K}$MajorCity(x, Ohio)], the sentence that states that the KB thinks it knows all the major cities of Ohio. What we do *not* want to conclude from the fact that the KB has an opinion about every city is that the sentence

$$\boldsymbol{K}\text{MajorCity(FavoriteCity(Joe), Ohio)} \vee \boldsymbol{K}\,\text{—MajorCity(FavoriteCity(Joe), Ohio)}$$

is true. This sentence claims that the KB has an opinion about the favorite city of Joe, which does not follow at all, given that the KB may have no idea about what city is being referred to. Of course, a KB *could* believe that Joe's favorite city is not a major city of Ohio without having identified it. For example, it might know that Joe only likes California cities. The point here is that a KB *need not* have an opinion either way. The only time we really want to conclude that

$$\boldsymbol{K}\text{MajorCity}(t, \text{Ohio}) \vee \boldsymbol{K}\,\text{—MajorCity}(t, \text{Ohio})$$

from the above is when the KB knows the referent of the term t (i.e., has identified the equivalence class to which it belongs).

The problem is that the sentence about Joe's favorite city is a direct consequence of the first one by the Axiom of Specialization. To remedy this, we have to replace the axiom (and its occurrence within Axiom KAX) by a special version (called AS*) that has an attached proviso: if $\forall x\alpha$ is true, then α_t^x is also true *provided no function symbol of t gets placed within the scope of a* $\boldsymbol{K}$ *in the substitution.* The motivation behind this proviso is as follows. The language $\mathscr{L}$ has the property that if two terms are equal, then they can be used interchangeably without affecting the truth value of any sentence. The language $\mathscr{KL}$, on the other hand, does not have this property. A sentence α_k^x may be true for every parameter k and yet be false for some term t (even though t has to be equal to one of the parameters). In particular, the sentence

$$(t_1 = t_2) \supset \boldsymbol{K}(t_1 = t_2)$$

is not a valid sentence of $\mathscr{KL}$ for every pair of terms. However, the sentence

$$\forall x \forall y(x = y) \supset \boldsymbol{K}(x = y)$$

is valid (and a theorem of $\mathscr{KL}$). So a KB characterized by the language $\mathscr{KL}$ has the interesting property that identicals are indiscernable for it (if two things are identical, the KB cannot help but know this) and yet there are identity statements that are neither known to be true nor known to be false.

As for the other axioms of $\mathscr{KL}$, it is worth noticing that the KCL axiom

corresponds to the Assumption of Closure (since it states that a pure sentence is known exactly when it is true) and that KAX, KMP and KUG together correspond to the Assumption of Competence. KAX says that the axioms of $\mathscr{L}$ are known while KMP and KUG say that the KB can perform Modus Ponens and Universal Generalization based on what it knows. If we were ever to consider a less competent version of a KB, these axioms would be the ones to modify.

The robustness of our approach is suggested by all the desirable properties of $\mathscr{KL}$ that are not explicit in its formalization. For example, the assumption that a KB need not have accurate knowledge of reality is reflected in the fact that the sentence

$$(\boldsymbol{K}\alpha \supset \alpha)$$

is not a theorem of $\mathscr{KL}$ unless α is pure, unlike most modal logics. For example, unlike the formalizations of knowledge in [10, 11], the set $\{\text{—}\phi(c), \boldsymbol{K}\phi(c)\}$ can be satisfied given a world and a KB that happens to be mistaken about that world. The sentence

$$\boldsymbol{K}(\boldsymbol{K}\alpha \supset \alpha),$$

however, *is* indeed a theorem, meaning that although a KB need not have accurate knowledge, it will always believe that it does. A KB that did not have this belief would have no way of deciding if what it was told by a user was indeed true. But the kind of KB we are considering is *committed* to its knowledge, since it always knows that what it knows is true.

One possible source of concern in our definition of $\mathscr{KL}$ is that by virtue of KCL and the fact that KCL is itself pure, the language may be *reducible*, in the sense that for any sentence using nested $\boldsymbol{K}$ operators, there would be another logically equivalent one that had no nested operators. The effect of this would be to say that meta-knowledge was superfluous in $\mathscr{KL}$, that is, that any sentence that spoke about the the meta-knowledge of a KB could be rephrased in terms of world knowledge. If that were the case, our concept of meta-knowledge would be somewhat vacuous and a lot of the machinery used in the definition of $\mathscr{KL}$ would not be necessary. However, despite the presence of KCL, it can be shown (by an argument similar to one appearing in [7]) that the language $\mathscr{KL}$ is *not* reducible in this sense. Intuitively, it should be clear that the sentence

$$\boldsymbol{K}[\exists x(\phi(x) \wedge \text{—}\boldsymbol{K}\phi(x))]$$

is not equivalent to any sentence without nested $\boldsymbol{K}$-operators. To prove this formally involves finding two KB structures on which the sentence has different

truth values and yet which know exactly the same set of sentences of $\mathscr{L}$. It follows that any sentence without nested $\boldsymbol{K}$-operators will have the same truth value on both KB structures and so cannot be equivalent to the above one.

On the other hand, it is not as if reducibility was an implausible notion since there is also a proof in [7] that a *propositional* version of $\mathscr{KL}$ is indeed reducible. So, in the propositional case, the concept of meta-knowledge is trivial and dispensable. As a result of this simplification, we could limit ourselves to a language where no $\boldsymbol{K}$-operator occurs in the scope of another. Moreover, in this case, we could interpret a sentence $\boldsymbol{K}\alpha$ as if it said that α was logically implied (in the standard propositional sense) by the KB. This simplifying interpretation is not possible for the full quantificational language, however.

2.2.5. *The definition of* TELL *and* ASK

Having considered the semantics of $\mathscr{KL}$, we can now define the two operations TELL and ASK. The functionality of the two operations will be

$$\begin{aligned} &\text{TELL: } \mathrm{KB} \times \mathscr{KL} \to \mathrm{KB}\ ; \\ &\text{ASK: } \mathrm{KB} \times \mathscr{KL} \to \{\text{yes, no, unknown}\}\ . \end{aligned}$$

The first argument to both operations is to be understood as an *abstract knowledge base*, that is, a partial picture of a world, specified by a set of world structures. The second argument to TELL is an *assertion*; the second argument to ASK is a *yes–no question.*

Given our semantic specification of $\mathscr{KL}$, we can define ASK easily since a KB should answer *yes* to a sentence of $\mathscr{KL}$ (used as a question) exactly when it knows that the sentence is true. More formally, we have the following.

Definition. ASK:

$$\mathrm{ASK}[\![k, \alpha]\!] = \begin{cases} \text{yes}, & \text{if } \boldsymbol{K}\alpha \text{ is true on } k; \\ \text{no}, & \text{if } \boldsymbol{K} \neg\alpha \text{ is true on } k; \\ \text{unknown}, & \text{otherwise}\ . \end{cases}$$

Notice that a negative answer, indicates that the KB thinks that the sentence is false, not that it does not know (in which case it would have answered *unknown*). Moreover, by the Assumption of Closure, no pure question will be answered *unknown.* In fact, a question will be answered *unknown* only if there is an element of k on which the assertion is true and another on which it is false. For example, if we let k_0 denote the set of all world structures, then k_0 specifies the KB with the least amount of world knowledge since any sentence

of $\mathscr{L}$ is *unknown* except the valid ones and their negations. On the other hand, there are sentences of $\mathscr{KL}$ other than the pure or valid ones for which k_0 will answer *yes*. The sentence

$$\exists x\phi(x) \supset \exists x(\phi(x) \wedge \neg \boldsymbol{K}\phi(x)),$$

for instance, is known to be true by k_0. This sentence says that if there is a ϕ, then there must be a ϕ that is not a known ϕ (since currently, there are none).[18] The set of all sentences α such that ASK$[\![k_0, \alpha]\!]$ is *yes* is *undecidable*. In fact, it is not even semi-decidable. The easiest way to see this is to notice that for σ in $\mathscr{L}$, $\neg \boldsymbol{K} \neg \sigma$ is in the set iff σ is satisfiable and that the set of satisfiable sentences of $\mathscr{L}$ is not recursively enumerable. So there can be no effective procedure that handles yes–no questions in all cases. The above definition of ASK presumes that any implementation will have a heuristic component.

The definition of TELL is more complicated, as far as sentences containing $\boldsymbol{K}$ are concerned. For sentences of $\mathscr{L}$, however, we want the result of telling a KB that α is true to be a KB that knows α and everything else it knew about the world, but no more than that. In other words, the resulting KB should only contain world structures $\langle s, v\rangle$ on which α is true (in the first-order sense defined earlier). This suggests that for sentences of $\mathscr{L}$, we have that

$$\text{TELL}[\![k, \alpha]\!] = k \cap \{\langle s, v\rangle \mid \alpha \text{ is true on } s \text{ and } v\}.$$

The result of telling a KB specified by k that α is true is described by the largest set of world structures (has least amount of world knowledge) that is both a subset of k (knows everything it did before) and a subset of the world structures satisfying α (knows that α is true).

As discussed above, the motivation behind using $\mathscr{KL}$ as an assertion language is to be able to tell the KB sentences like

$$\exists x(\phi(x) \wedge \neg \boldsymbol{K}\phi(x)).$$

The interpretation of this assertion is not that there is a ϕ that will *never* be known, but that there is a ϕ that is not *currently* known. So within the context of a TELL, instances of $\boldsymbol{K}\alpha[x_1, \ldots, x_n]$ should be understood as the α's that were known *just prior to the* TELL *operation*. With this understanding, we can define TELL for all sentences of $\mathscr{KL}$ as follows:

[18]The fact that this sentence is known by the KB with the least amount of world knowledge is not really a property of the *logic* of $\mathscr{KL}$. From the point of view of the semantics, there is nothing special about this sentence other than it being satisfiable. It is the behavior of ASK with respect to specific KB structures that leads to this observation.

Definition. TELL:

$$\text{TELL}[\![k, \alpha]\!] = k \cap \{\langle s, v\rangle \mid \alpha \text{ is true on } s, v \text{ and } k\}.$$

This definition specifies that the result of a TELL on a KB described by k is just those elements of k where α is true, understanding any reference in α to what is known to mean known by the KB specified by k. This has the effect, as desired, that any pure assertion either produces k itself (when it is redundant) or the empty set (when it is contradictory). In general, the only assertions that are not either redundant or contradictory are those for which ASK would return *unknown* and, by the Assumption of Closure, no pure assertion is *unknown*.

One thing to notice about this definition of TELL and ASK is that the world knowledge of a KB is *monotonic* but that in general, knowledge is *non-monotonic*. As knowledge is acquired, we are assuming an ever more refined picture of the world in that any known sentence of $\mathscr{L}$ remains known.[19] But this is not true of $\mathscr{KL}$; not every known sentence of $\mathscr{KL}$ will remain known as knowledge is acquired. For example, as pointed out above,

$$\text{ASK}[\![k_0, [\exists x \phi(x) \supset \exists x(\phi(x) \wedge \neg \mathbf{K}\phi(x))]]\!] = \text{yes}$$

and yet

$$\text{ASK}[\![\text{TELL}[\![k_0, \phi(1)]\!], [\exists x \phi(x) \supset \exists x(\phi(x) \wedge \neg \mathbf{K}\phi(x))]]\!] = \text{unknown}.$$

In other words, after telling k_0 that *1* is a ϕ, the KB no longer believes that if there is a ϕ, then there has to be one it does not know about. Furthermore,

$$\text{ASK}[\![\text{TELL}[\![k_0, \forall x[\phi(x) \equiv (x = 1)]]\!],$$
$$[\exists x \phi(x) \supset \exists x(\phi(x) \wedge \neg \mathbf{K}\phi(x))]]\!] = \text{no},$$

so that after telling k_0 that *1* is the *only* ϕ, it now believes that the original sentence is false. So while answers to questions in $\mathscr{L}$ can only move from *unknown* to *yes* or *no* as knowledge is acquired, answers to questions in $\mathscr{KL}$ can change arbitrarily as knowledge is acquired. Notice, however, that this non-monotonicity is not a property of the logic (proof-theory or semantics) of $\mathscr{KL}$, but of its *pragmatics*, which is perhaps as it should be [12].

[19]Essentially what this says is that TELL is not the appropriate operation for *belief revision* in the sense of tracking down a belief to be discarded in the presence of contradictory information. This is a thorny problem given the expressive power of $\mathscr{L}$. For example, if a KB is told $(\alpha \vee \beta)$, then $\neg\alpha$ and then $\neg\beta$, it is far from clear how its beliefs should be revised. The position here is that the competence of a KB should at the very least be able to detect the inconsistency as the first step towards making an intelligent decision about how to handle the conflict.

3. The Symbol Level

In the previous section, we considered an extremely abstract notion of a KB, specified by a set of world structures. We also examined how the TELL and ASK operations could be defined over such sets in a way that made no assumptions about how the knowledge was represented. A number of questions remain unanswered, however. First of all, exactly what sets of world structures do indeed specify a KB? Does every distinct set of worlds constitute a distinct body of world knowledge? Can this world knowledge always be represented symbolically? Given a symbolic representation of the knowledge in a KB, how can the TELL and ASK operations be defined over these representations. These are the questions that will be answered in this section. So the purpose of the section is to solidify the notion of a KB and show that, despite the presence of $\mathcal{KL}$ as an assertion language, the kind of KB we are considering is one that can be represented in first-order terms.

3.1. The representation of knowledge

In this section, we examine closely sets of world structures and determine which of these can be said to specify knowledge bases. In so doing we will isolate two special categories of KB structures: the *representable* and the *ω-representable* sets of world structures. The problem we have to face is that only the first kind can be represented symbolically, but the language $\mathcal{KL}$ forces us to also deal with the second.

3.1.1. *Representable KB structures*

In the preceding section, we defined TELL and ASK at the knowledge level independently of any symbolic representation of knowledge. However, we made no special provisions to ensure that the kind of knowledge we were considering was indeed representable in first-order terms at the symbol level. The kind of representation we have in mind here, obviously, is using a sentence (or a finite set of sentences) of $\mathcal{L}$ to refer to a set of worlds, that is, using a finite collection of symbols (instead of a possibly uncountably infinite set of world structures) to represent what is known. Thus, a sentence of $\mathcal{L}$ *represents* those worlds that satisfy the sentence. We will call a KB structure k *representable* iff there is a sentence of $\mathcal{L}$ such that k is the set of world structures satisfying that sentence. Clearly, our first concern in $\mathcal{KL}$ must be with a semantics ranging over representable KB structures since these are those for which the KB can be realized. However, as we will show below, we must be careful in restricting our attention to just these sets.

The first thing to notice is that not every KB structure is representable. Consider, for example, an arbitrary ordering of the parameters i_1, i_2, and so on.

Now let

$$\text{ODD} = \{i_1, i_3, \ldots\}, \quad \text{every second element of the ordering}.$$

Finally, assume that ϕ is some monadic predicate and consider the set

$$\{\langle s, v\rangle \mid \phi(i) \text{ is true on } s \text{ and } v \text{ iff } i \in \text{ODD}\}$$

This set of world structures is for a KB that knows that the ϕ are the same as the ODD-parameters. Because no single sentence of $\mathscr{L}$ can represent that knowledge about ϕ, the KB is not representable.

Now consider the following sentence of $\mathscr{KL}$:

$$\begin{aligned}
&\forall x \neg\theta(x, x) \\
&\quad \wedge \forall x \forall y \forall z[\theta(x, y) \wedge \theta(y, z) \supset \theta(x, z)] \\
&\quad \wedge \boldsymbol{K}\phi(1) \wedge \boldsymbol{K} \neg\phi(2) \\
&\quad \wedge \forall x[\boldsymbol{K}\phi(x) \supset \exists y\theta(x, y) \wedge \boldsymbol{K}\phi(y)] \\
&\quad \wedge \forall x[\boldsymbol{K} \neg\phi(x) \supset \exists y\theta(x, y) \wedge \boldsymbol{K} \neg\phi(y)].
\end{aligned}$$

This sentence first says that θ is irreflexive and transitive (think of it as 'less than' where parameters are treated as numbers). Next, it says that there is a known ϕ and a known non-ϕ. Finally, it says that for each known ϕ there is a 'larger' one and similarly for the known non-ϕ. So the sentence implies that the known ϕ and the known non-ϕ are both infinite in number. This sentence is satisfiable since it is true in a world where θ is interpreted as the *less-than* relation and the KB is the one that knows about the ODD-parameters, above. However, the crucial point is that this sentence is not satisfied by *any* representable KB structure.

This leaves us in somewhat of a quandary. The representable sets are defined to be the *only* KB structures that can be represented at the symbol level. The language $\mathscr{KL}$, on the other hand, forces us to consider non-representable sets. If we imagine the semantics of $\mathscr{KL}$ adjusted so that only representable KB structures were used, the negation of the above sentence would have to be valid (and a theorem). In other words, either we change the semantics (and proof theory) of $\mathscr{KL}$ or we admit that there are knowledge bases that cannot be represented symbolically.

It is very difficult to imagine how the proof theory of $\mathscr{KL}$ could be modified to make sentences like the negation of the above come out as theorems. One can even imagine proving that no *axiomatic* theory would be up to the task.[20]

[20]As will become clear below, a theory that allows all representable KB structures without admitting any non-representable ones is like a number theory that says that numbers can be arbitrarily large and yet somehow rules out infinitely large ones. It seems beyond the power of recursively axiomatizable theories in $\mathscr{L}$ or $\mathscr{KL}$ to capture such distinctions.

So we are almost forced into looking beyond representable sets to see exactly what sort of KB we can, in fact, describe using sentences of $\mathcal{KL}$.

3.1.2. *ω-representable KB structures*

The non-representable KB structure above can be thought of as specifying one that knows

$$\{\phi(i_1), \neg\phi(i_2), \phi(i_3), \neg\phi(i_4), \ldots\}.$$

This suggests a generalization of representable sets that allows the knowledge to be represented using a potentially *infinite* set of sentences of $\mathcal{L}$. We will call a KB structure k *ω-representable* iff there is a set of sentences of $\mathcal{L}$ such that k is the set of world structures satisfying the set. Obviously, every representable set is ω-representable but not vice versa.

The ω-representable KB structures turn out to be just the right generalization of representable ones. For example, we might say that two knowledge bases are *equivalent* iff they know exactly the same sentences of $\mathcal{L}$. Then, no two distinct ω-representable sets are equivalent. Moreover, it is easy to verify that for *any* KB structure, there is a unique ω-representable one that is equivalent to it. So we can partition the KB structures into equivalence classes (with respect to the above definition of equivalence) and find exactly one ω-representable KB structure in each class.

Consider, for example k_0, the set of all world structures, a ω-representable set represented by, among others, the empty set of sentences. It is shown in [7] that the set k_0 and the set formed by removing any single world structure from k_0 know exactly the same sentences of $\mathcal{L}$. In other words, the two KB structures are equivalent. The question is why we would ever want to consider there to be *two* sets here in the first place; semantically, we would like to be able to say that there is only a *single* abstract body of knowledge under consideration, and therefore, a single set of worlds. This can be done very simply by restricting our attention to ω-representable sets. In this way we never get two distinct KB structures unless there is a sentence of $\mathcal{L}$ that they disagree on. So ω-representable sets are exactly analogous to world structures: just as for any two distinct world structures there is a sentence of $\mathcal{L}$ that one says is *true* and the other not, for any two distinct ω-representable KB structures, there is a sentence of $\mathcal{L}$ that one says is *known* and the other not.

The key property of ω-representable sets, however, is that as far as $\mathcal{KL}$ is concerned, we can restrict our attention to them without any loss of generality. In [7], the following theorem is proven.

Maximal Set Theorem.[21] *If a sentence of $\mathcal{KL}$ is satisfiable at all, it is satisfied on some ω-representable KB structure.*

In other words, restricting our attention to ω-representable sets does not change the semantics of $\mathcal{KL}$. Exactly the same sentences are valid and satisfiable. In particular, any (finite) condition we can impose on a KB using a sentence of $\mathcal{KL}$ can be satisfied by one specified by a ω-representable set, if at all.[22] This is not a trivial result, however, in that given an arbitrary KB structure, the ω-representable set predicted by the theorem will not necessarily be equivalent to it (that is, have the same world knowledge). Rather, the set is the result of an elaborate construction where a KB has to be given world knowledge about an *infinite* number of predicates not appearing in the sentence.

Despite how naturally the ω-representable sets fit into the semantics of $\mathcal{KL}$, they do have a very serious drawback. Specifically, the TELL function does not necessarily map one ω-representable set into another. For instance, let

$$k = \{\langle s, v\rangle \mid \text{for every } i \in \text{ODD}, \phi(i) \text{ is true on } s \text{ and } v\}.$$

So, k specifies a KB that knows that every element of ODD has the property ϕ. It is ω-representable and is represented by

$$\{\phi(i) \mid i \in \text{ODD}\}.$$

If we now let

$$k' = \text{TELL}[\![k, [\exists x(\psi(x) \wedge \mathbf{K}\phi(x))]]\!],$$

then we have that k' is the result of telling k that there is a ψ among the known ϕ. Since the parameters that are known to be ϕ are the ODD ones, k has been told that there is a ψ among the ODD-parameters. In other words, k' is the set

$$\{\langle s, v\rangle \mid \text{for every } i \in \text{ODD}, \phi(i) \text{ is true on } s \text{ and } v \\ \text{and for some } i \in \text{ODD}, \psi(i) \text{ is true on } s \text{ and } v\}.$$

[21] In [7], ω-representable sets are called 'maximal' and are defined as follows: a KB structure k is *maximal* iff it is the set of all $\langle s, v\rangle$ such that $(\mathbf{K}\alpha \supset \alpha)$ is true on s, v and k, for every α in $\mathcal{L}$. This definition can be shown to be equivalent to the current one.

[22] There are, however, *infinite* sets of sentences that are satisfiable but not by any ω-representable KB structure. One such is the set of all pure sentences that are true on k', described below. If the expressive power of a language is taken to be a function of what distinctions *sentences* can make, the move to ω-representable sets is without loss of generality; but if it is a function of what *sets of sentences* can do, the restriction is real.

Thus, k' believes that every ODD-parameter is ϕ and that at least one is ψ.

The KB structure k' is anomalous, however. First of all, it is not ω-representable. It can be shown that the ω-representable set that is equivalent to it (call it k'_*) contains a world structure where ψ is true for no ODD-parameter. In other words, k'_* no longer believes that there is an ODD ψ as k' did. And yet, k' and k'_* are equivalent since they agree completely on every sentence of $\mathscr{L}$. Where they disagree is on the sentence

$$\exists x(\psi(x) \wedge \boldsymbol{K}\phi(x)).$$

The KB described by k' believes that it is true while the one described by k'_* does not.

It might be thought that the restriction to ω-representable sets was perhaps too facile, that k' and k'_* in fact specify knowledge bases that must be considered distinct. Perhaps all of $\mathscr{KL}$ must be used to represent world knowledge (and to decide whether two KBs have identical world knowledge). This would allow k' and k'_* to be distinguished in terms of the sentence of $\mathscr{KL}$ on which they disagree.

As it turns out, however, it is k' itself that is at fault here and a generalization that allows all of $\mathscr{KL}$ to be used to represent knowledge would not solve the problem. To see this, consider

$$k'' = \text{TELL}[\![k', [\forall x\phi(x)]]\!].$$

This KB knows that every parameter (not just the ODD ones) is a ϕ. Assuming that its knowledge about the ψ is unchanged by this assertion, k'' cannot just believe that there is a ψ among the known ϕ since it would then lose all connection with the ODD-parameters. But what exactly does the KB described by k'' believe about ψ? This KB is truly anomalous since we can no longer represent its belief *even using an infinite set of $\mathscr{KL}$ sentences.*[23] In some sense, it was coincidental that we could represent the knowledge of k': we characterized what it knew about ψ using what it knew about ϕ. But even if we consider representing the knowledge of k' using sentences of $\mathscr{KL}$, we will still run into situations (like k'') where even $\mathscr{KL}$ is no help. The conclusion: k' is anomalous and, therefore, not every ω-representable KB structure can be used as an argument to TELL.

[23]The best we might do is use an infinitely long disjunction $[\psi(i_1) \vee \psi(i_3) \vee \cdots]$ or somehow time stamp a KB to be able to refer to the ϕ that were known just prior to the assertion.

So where we have arrived in our examination of representable and ω-representable sets is the following.

(1) The language $\mathscr{KL}$ forces us into considering knowledge bases that are not representable.

(2) The KB's described by ω-representable KB structures are both necessary and sufficient from the point of view of $\mathscr{KL}$, but are not closed under the TELL operation.

So what we have to do[24] is try to understand ω-representable sets in such a way that the knowledge bases they specify cannot be used as arguments to TELL. One account that does the trick is to understand these sets as limiting cases of representable ones.

Let $k_1, k_2, \ldots$ be an infinite sequence of representable KB structures where each element in the sequence (as a set of world structures) is a subset of the previous one (that is, a monotone decreasing sequence). We call a KB structure k_ω the *limit of the sequence* if it is the largest set that is a subset of every element in the sequence (that is, it is the greatest lower bound). It is easy to show that the ω-representable sets are precisely the limits of descending sequences of representable sets. This kind of sequence is, in fact, a path of *knowledge acquisition* since each element in the sequence must know at least as much as its predecessors. So a representable set specifies a KB that is located on a knowledge acquisition path while a ω-representable set specifies the ultimate destination of such a path, which may never be attained. For example, if k_ω is the ω-representable set represented by some set of sentences $\{\alpha_i\}$, then it is also the limit of a sequence of representable sets

$$
\begin{aligned}
&k_0 = \text{the set of all world structures},\\
&k_1 = \text{TELL}[\![k_0, \alpha_1]\!],\\
&k_2 = \text{TELL}[\![k_1, \alpha_2]\!],\\
&k_3 = \text{TELL}[\![k_2, \alpha_3]\!],\\
&\vdots
\end{aligned}
$$

In other words, to be the limit of an infinite descending sequence of representable sets is also to be the limit of an infinite set of TELL operations. So the reason a KB described by a non-representable set cannot be told anything is that *you never really get there* except in the limit. In particular, although it makes sense to consider the limit of a sequence of operations, it does not make sense to *arrive* at the limit point and perform yet another operation. Although the language $\mathscr{KL}$ forces us to admit the existence of these limit points, we can understand TELL and ASK as dealing only with the representable ones.

[24]Another option we have not considered is to try a notion of representability somewhere between the representable and ω-representable sets. For example, we might consider a KB that can be represented by a *recursive* or *recursively enumerable* set of sentences, although it is unlikely that we could get by without modifying the semantics for $\mathscr{KL}$.

The only thing left to do to with representable sets is to establish that (unlike ω-representable sets) they are closed under the TELL operation. If the only assertions under consideration were sentences of $\mathscr{L}$, this would be easy to show since the result of telling a KB k that α is true (where α is first-order) could be represented by $(k \wedge \alpha)$. The fact that we need only consider representable KB structures *even when assertions are taken from* $\mathscr{KL}$ is the major result of this research, which we now describe.

3.2. The representation theorem

It is perhaps worthwhile after the excursion into the mathematical properties of $\mathscr{KL}$, TELL and ASK and KB structures, to reconsider what has been accomplished and where we stand.

We started with a notion of a KB as a first-order sentence and Tell- and Ask-operations that were purely proof theoretic: Tell simply conjoined the assertion to the KB and Ask used an oracle for first-order theoremhood to determine its answer. We then replaced the symbolic notion of a KB by a functional (representation independent) one: a KB was thought of as anything having a certain 'world knowledge', understood as a certain behavior under the interaction operations. We showed how a first-order interaction language was too weak and subsequently replaced it with a more expressive one, $\mathscr{KL}$. In the process of defining the TELL and ASK operations in terms of the semantics of $\mathscr{KL}$, we were lead to an abstract picture of a KB as anything embodying a partial picture of a world, and therefore specifiable by a set of world structures. We showed that without loss of generality we could restrict our attention to the ω-representable sets of world structures and that a subset of these, the representable sets, specified knowledge bases that could be represented using sentences of $\mathscr{L}$.

The question that now remains is the following: if we chose to represent a KB using a sentence of $\mathscr{L}$, what can be said about the TELL and ASK operations when $\mathscr{KL}$ is the language of interaction? Specifically:

(1) Is the KB after a TELL operation always representable and how might a representation be constructed from the representation before the TELL?

(2) How is the ASK operation when using a sentence of $\mathscr{KL}$ related to the ASK operation over first-order sentences?

The answer here is somewhat surprising (and will be examined in detail): despite the increase in expressive power given by $\mathscr{KL}$,

(1) The result of a TELL can always be represented by conjoining some first-order sentence to the KB.

(2) Given a first order KB, the ASK operation can be performed using only the oracle for first-order theoremhood.

So even though we are communicating with a KB in an intensional language that is more powerful than $\mathscr{L}$, this communication can be understood and

represented completely in first-order terms. So for example, one could imagine using a standard (first-order) resolution theorem prover to implement the ASK operation.[25] A version of TELL and ASK that uses this oracle will be defined below with the understanding that the definition is not intended to demonstrate an *efficient* algorithm, only that an algorithm is *possible*, given the oracle.

3.2.1. *Method*

The main observation leading to the Representation Theorem is that given a first-order sentence k that represents some world knowledge, it is possible to *reduce* any formula α of $\mathscr{KL}$ to a first-order formula which we will call $|\alpha|_k$, without changing its assertional force. If α has n free variables, then the result of telling the KB that (or asking the KB if) $|\alpha|_k[i_1, \ldots, i_n]$ is true is the same as if $\alpha[i_1, \ldots, i_n]$ had been used. Notice that $|\alpha|_k$ is indexed by the KB k and so the formula will change as the KB changes. In fact, one of the properties of $\mathscr{KL}$ is that it is impossible to find a first order sentence that is equivalent to α for *every* KB.

As it turns out, being able to handle the general case of $|\alpha|_k$ depends only on a correct treatment of $|\boldsymbol{K}\alpha|_k$ when α is a formula of $\mathscr{L}$. To see an example of how this will work, imagine that k is the sentence

$$\text{City(LosAngeles)}$$

where, for this example, constants are treated like parameters (that is, Los Angeles is the only known city). Now suppose we want a first-order equivalent to $\boldsymbol{K}\text{City}(x)$ for this KB. That is to say, we want a formula of $\mathscr{L}$, $|\boldsymbol{K}\text{City}(x)|_k$, with one free variable x, such that for any parameter i,

$$(|\boldsymbol{K}\text{City}(x)|_k)^x_i \text{ is true} \quad \text{iff} \quad \boldsymbol{K}\text{City}(i) \text{ is true}.$$

But this sentence is true for the KB exactly when i is the LosAngeles parameter. This suggests that $|\boldsymbol{K}\text{City}(x)|_k$ can be the formula $(x = \text{LosAngeles})$, instances of which are *valid* when x is LosAngeles and *unsatisfiable* otherwise. On the other hand, had k been

$$\text{City(LosAngeles)} \wedge \text{City(NewYorkCity)} \wedge [\text{City(Boston)} \vee \text{City(Detroit)}],$$

the formula would have been $[(x = \text{LosAngeles}) \vee (x = \text{NewYorkCity})]$ since neither Boston nor Detroit are known instances of City (although it is known

[25]This assumes that we could define a theorem prover to work with $\mathscr{L}$, a slightly non-standard first-order dialect. Indeed, the treatment of *equality* in $\mathscr{L}$ is critical to this approach.

that one of them is a city). Similarly, $|\boldsymbol{K}\text{City}(z)|_k$ when k is $\forall x\text{City}(x)$ is $(z = z)$ since every parameter is a known city in this case.

To simplify our notation, we will let RES$[\![k, \alpha]\!]$ stand for $|\boldsymbol{K}\alpha|_k$ when α is a formula of $\mathscr{L}$. So, for example, RES$[\![$City(LosAngeles), City$(z)]\!]$ is $(z =$ LosAngeles). The situation becomes more interesting when the formula we want to reduce contains multiple free variables. For example, if k is the sentence $[\psi(1, 7) \wedge \psi(5, 2)]$, then RES$[\![k, \psi(5, y)]\!]$ is $(y = 2)$ while RES$[\![k, \psi(x, y)]\!]$ is

$$(x = 1 \wedge y = 7) \vee (x = 5 \wedge y = 2).$$

It may not always be possible just to 'read' off the correct values for the free variables. For example,

$$\text{RES}[\![\forall x \forall y[x = y \equiv \neg\psi(y, x)], \psi(x, 4)]\!] = (x \neq 4)$$

and

$$\text{RES}[\![(\psi(1, 6) \wedge \forall x \forall y[[(x = y) \vee \psi(x, y)] \supset \psi(y, x)]), \psi(6, y)]\!] = [(y = 1) \vee (y = 6)].$$

In general, calculating RES$[\![k, \alpha]\!]$ may require arbitrary amounts of deduction and there will be no effective procedure for calculating it.

Before proceeding to formally define a RES$[\![k, \alpha]\!]$ and proving that it works for any KB, we will simply assume that it works and show how it leads to the Representation Theorem. First of all, we can define the general case of $|\alpha|_k$ in terms of RES$[\![k, \alpha]\!]$ as follows:

$$|\alpha|_k = \alpha \quad \text{when } \alpha \in \mathscr{L},$$
$$|\neg\alpha|_k = \neg|\alpha|_k,$$
$$|\alpha \supset \beta|_k = (|\alpha|_k \supset |\beta|_k),$$
$$|(\forall x)\alpha|_k = (\forall x)|\alpha|_k,$$
$$|\boldsymbol{K}\alpha|_k = \text{RES}[\![k, |\alpha|_k]\!].$$

This gives us $|\alpha|_k$ for any α in $\mathscr{KL}$ in terms of how it handles $\boldsymbol{K}\alpha$ when α is first-order. For example,

$$|\forall x(\text{City}(x) \supset \boldsymbol{K}\text{City}(x))|_{\text{City(LosAngeles)}} = \forall x(\text{City}(x) \supset x = \text{LosAngeles}).$$

What this says is that if all that is known is that Los Angeles is a city, then the sentence that says that every city is known reduces to one that says that every city is Los Angeles (that is, that it is the only city).

To establish more rigorously the fact that $|\alpha|_k$ correctly captures α given that k represents everything that is known, we have to be clear about what knowledge a first-order sentence k represents. So, we define

$$\mathscr{R}[k] = \{\langle s, v\rangle \mid k \text{ is true on } s \text{ and } v\}.$$

A first-order sentence k presents a picture of a world specified by those world structures on which the sentence is true. In other words, a sentence of $\mathscr{L}$ is taken to represent the knowledge that the world satisfies it. The first observation about $\mathscr{R}[k]$ is the following.

Lemma 3.1. *For any sentences α and k of $\mathscr{L}$, $\boldsymbol{K}\alpha$ is true on $\mathscr{R}[k]$ iff $(k \supset \alpha)$ is a theorem of $\mathscr{L}$.*

Proof. The sentence $\boldsymbol{K}\alpha$ is true on $\mathscr{R}[k]$ when α is true on every element of $\mathscr{R}[k]$, that is, when every world structure satisfying k also satisfies α. Then, assuming the proof theory for $\mathscr{L}$ is complete, this happens exactly when $(k \supset \alpha)$ is a theorem of $\mathscr{L}$.

This lemma says that a first-order sentence is known exactly when it is a (first-order) consequence of the sentence used to represent the knowledge. This does not mean that we can define RES$[\![k, \alpha]\!]$ to be $(k \supset \alpha)$ since this sentence will not, in general, be *false* just because $\boldsymbol{K}\alpha$ is false. The above lemma only guarantees that, in this case, it will not be *valid*. However, what we will prove below (in Lemma 3.6) is that for any formula α of $\mathscr{L}$,

$$\begin{aligned} &\boldsymbol{K}\alpha[i_1, \ldots, i_n] \text{ is true on } \mathscr{R}[k] \\ &\quad \text{iff} \quad \text{RES}[\![k, \alpha]\!][i_1, \ldots, i_n] \text{ is true (on } \mathscr{R}[k]). \end{aligned}$$

Given this property, and our definition of $|\alpha|_k$ in terms of RES$[\![k, \alpha]\!]$, we can prove the following.

Lemma 3.2. *For any formula α of $\mathscr{KL}$, any k in $\mathscr{L}$, any set of parameters $i_1, \ldots, i_n$, and any world structure $\langle s, v\rangle$, $\alpha[i_1, \ldots, i_n]$ is true on s, v, and $\mathscr{R}[k]$ iff $|\alpha|_k[i_1, \ldots, i_n]$ is true on s and v.*

Proof. By induction on the structure of α. If α is a formula of $\mathscr{L}$, the lemma trivially holds, since $|\alpha|_k$ is α. Otherwise, assume the lemma holds for two arbitrary formulas α and β. It will then clearly hold for $\neg\alpha$ and $(\alpha \supset \beta)$. As for universal generalization, $\forall x\alpha[i_1, \ldots, i_n, x]$ is true iff for every parameter i, $\alpha[i_1, \ldots, i_n, i]$ is true iff (by induction) $|\alpha|_k[i_1, \ldots, i_n, i]$ is true iff $\forall x|\alpha|_k[i_1, \ldots, i_n, x]$ is true iff $|\forall x\alpha|_k[i_1, \ldots, i_n, x]$ is true. Finally, $\boldsymbol{K}\alpha[i_1, \ldots, i_n]$ is true on $\mathscr{R}[k]$ iff $\alpha[i_1, \ldots, i_n]$ is true on every world structure in $\mathscr{R}[k]$ iff (by induction) $|\alpha|_k[i_1, \ldots, i_n]$ is true on every world structure in $\mathscr{R}[k]$ iff $\boldsymbol{K}|\alpha|_k[i_1, \ldots, i_n]$ is true on $\mathscr{R}[k]$. But $|\alpha|_k$ is, by definition, a formula of $\mathscr{L}$. So, by Lemma 3.6 to be proven below, $\boldsymbol{K}|\alpha|_k[i_1, \ldots, i_n]$ is true on $\mathscr{R}[k]$ iff RES$[\![k, |\alpha|_k]\!][i_1, \ldots, i_n]$ is true. Thus, $\boldsymbol{K}\alpha[i_1, \ldots, i_n]$ is true iff $|\boldsymbol{K}\alpha|_k[i_1, \ldots, i_n]$ is true.

Lemma 3.2 establishes (given an appropriate definition of RES$[\![k, \alpha]\!]$) that $|\alpha|_k$ correctly captures within a first-order formula what α was saying about the KB. We can now prove the Representation Theorem which shows how TELL and ASK can be defined given k as the representation of what is known.

Representation Theorem. *For any KB k and any sentence of $\mathcal{KL}$ α,*

$$\text{TELL}[\![\mathcal{R}[k], \alpha]\!] = \mathcal{R}[k \wedge |\alpha|_k]$$

$$\text{ASK}[\![\mathcal{R}[k], \alpha]\!] = \begin{cases} \text{yes}, & \textit{if } \vdash (k \supset |\alpha|_k); \\ \text{no}, & \textit{if } \vdash (k \supset -|\alpha|_k); \\ \text{unknown}, & \textit{otherwise}. \end{cases}$$

Proof. For ASK$[\![\mathcal{R}[k], \alpha]\!]$, the answer will be *yes* precisely when $\boldsymbol{K}\alpha$ is true on $\mathcal{R}[k]$ which, by Lemma 3.2 coincides with $\boldsymbol{K}|\alpha|_k$ being true on $\mathcal{R}[k]$. But since $|\alpha|_k$ is first-order, by Lemma 3.1, this happens exactly when the sentence $(k \supset |\alpha|_k)$ is a theorem of $\mathcal{L}$. The situation is analogous when the answer from ASK is *no*. The proof for TELL, on the other hand, follows from Lemma 3.2 directly.

$$\begin{aligned} \text{TELL}[\![\mathcal{R}[k], \alpha]\!] &= \mathcal{R}[k] \cap \{\langle s, v\rangle \mid \alpha \text{ is true on } s, v \text{ and } \mathcal{R}[k]\} \\ &= \mathcal{R}[k] \cap \{\langle s, v\rangle \mid |\alpha|_k \text{ is true on } s, v\} \\ &= \{\langle s, v\rangle \mid k \text{ and } |\alpha|_k \text{ are true on } s, v\} \\ &= \mathcal{R}[k \wedge |\alpha|_k]. \end{aligned}$$

The Representation Theorem not only guarantees that the result of a TELL will be first-order representable no matter what the argument, it also stipulates that it will be representable by an extension of the existing KB (that is, by conjoining something to it). Moreover, the Representation Theorem says that, no matter what the argument, the result of an ASK depends only on whether or not a first-order sentence is implied by the KB. However, the theorem assumes a very specific but yet to be proven property of RES$[\![k, \alpha]\!]$, which is that

$\boldsymbol{K}\alpha[i_1, \ldots, i_n]$ is true on $\mathcal{R}[k]$
iff RES$[\![k, \alpha]\!][i_1, \ldots, i_n]$ is true (on $\mathcal{R}[k]$).

Once the definition of RES$[\![k, \alpha]\!]$ is provided below and the critical Lemma 3.6 is proven, it will be clear that the Representation Theorem not only provides a proof theoretic view of the TELL and ASK operations, but a *first-order* proof theoretic view.

The reason this theorem is so important is that it forms a bridge not only between the knowledge level and symbol level but between $\mathcal{KL}$ and $\mathcal{L}$. It says that we can use $\mathcal{KL}$ to handle assertions like closed (or open) world assumptions and defaults (as we will discuss later), but still restrict our implementation

methods to standard first-order techniques (such as resolution theorem proving). Not that this makes TELL or ASK decidable in the general case; in fact, since both need to calculate $|\alpha|_k$ when α is not first order, both are undecidable. But the only oracle we need is one that can tell us if $(k \supset \alpha)$ is a first-order theorem (where both sentences are first-order). In other words, provided we use $\mathscr{L}$ as our symbolic representation language, this oracle is all we need to correctly implement a TELL and ASK operation for any sentence of $\mathscr{KL}$.

3.2.2. *Proof*

The definition of RES$[\![k, \alpha]\!]$ that we will use is by no means the shortest formula that will work, as it uses every parameter appearing in either in k or α. The definition is

RES$[\![k, \alpha]\!]$ = **If** α has no free variables
then if $\vdash (k \supset \alpha)$
then $\forall x(x = x)$
else $\forall x(x \neq x)$
else (Assume that x is free in α and that the parameters appearing in k or α are $i_1, \ldots, i_n$. Let i be the "least" parameter not in k or α.)

$$((x = i_1) \wedge \text{RES}[\![k, \alpha^x_{i_1}]\!]) \vee \cdots \vee ((x = i_n) \wedge \text{RES}[\![k, \alpha^x_{i_n}]\!]) \vee ((x \neq i_1) \wedge \cdots \wedge (x \neq i_n) \wedge \text{RES}[\![k, \alpha^x_i]\!]^i_x).$$

So, if α is a sentence, then RES$[\![k, \alpha]\!]$ is either valid or inconsistent depending on whether or not α is implied by k. Otherwise, assuming that x is a free variable of α, we call RES$[\![k, \alpha]\!]$ recursively with one fewer free variable by replacing x in α by some parameter. First, we use the parameters appearing in α or k, and then we use another parameter, i, that does not. The first thing to establish here is that the choice of i is irrelevant. To do so, we will show that a theorem of $\mathscr{L}$ remains a theorem when the parameters in it are renamed consistently.

Lemma 3.3. *Let $*$ be a bijection over parameters and, for any α in $\mathscr{L}$, let α^* be the result of replacing each i in α by i^*. Then, α is a theorem iff α^* is.*

Proof. By induction on the length of the proof of α. If α is an axiom of $\mathscr{L}$, then there are two cases: if α is not an instance of the equality schema, then the presence of parameters in α is incidental and *any* renaming of the parameters is still an axiom; if, on the other hand, α is of the form $(i = i) \wedge (i \neq j)$ for distinct parameters i and j, then α^* is $(i^* = i^*) \wedge (i^* \neq j^*)$ which is also an instance of the equality schema since $*$ is a bijection. Now suppose that α

follows from $(\sigma \supset \alpha)$ and σ by Modus Ponens. By induction, σ^* and $(\sigma^* \supset \alpha^*)$ must also be theorems, so α^* follows again by Modus Ponens. Finally, suppose $\forall x\alpha$ follows by Universal Generalization from sentences α_i^x, for i ranging over all the parameters occurring in α and at least one not present in α. By induction, the sentences $(\alpha_i^x)^*$, that is, $\alpha^*{}_{i^*}^{x}$, must also be theorems. But since $*$ is a bijection, i appears in α iff i^* appears in α^*. So, the i^* must now range over all the parameters appearing in α^* and one that does not occur in α^*. Thus, by Universal Generalization, $\forall x\alpha^*$, that is, $(\forall x\alpha)^*$, is also a theorem.

Corollary 3.4. *If $*$ is a bijection and α is a formula such that α^* is α, then for any set of parameters, $\alpha[i_1, \ldots, i_n]$ is a theorem iff $\alpha[i_1^*, \ldots, i_n^*]$ is.*

Given this renaming property, we can show that $\mathrm{RES}[\![k, \alpha]\!]$ produces a formula whose instances are theorems exactly when instances of $(k \supset \alpha)$ are.

Lemma 3.5. *For any formula α and sentence k of $\mathscr{L}$, and any parameters $j_1, \ldots, j_m$, the sentence $\mathrm{RES}[\![k, \alpha]\!][j_1, \ldots, j_m]$ is a theorem iff $(k \supset \alpha)[j_1, \ldots, j_m]$ is a theorem.*

Proof. By induction on the number of free variables in α. If α is a sentence, then $\mathrm{RES}[\![k, \alpha]\!]$ is a theorem iff it is $\forall x(x = x)$ which happens only when $(k \supset \alpha)$ is a theorem. Otherwise, suppose α is $\alpha[x_1, \ldots, x_m, x]$. By definition, $\mathrm{RES}[\![k, \alpha]\!]$ is the formula

$$\begin{aligned}&((x = i_1) \wedge \mathrm{RES}[\![k, \alpha_{i_1}^x]\!])\\ &\quad \vee \cdots \vee ((x = i_n) \wedge \mathrm{RES}[\![k, \alpha_{i_n}^x]\!])\\ &\quad \vee ((x \neq i_1) \wedge \cdots \wedge (x \neq i_n) \wedge \mathrm{RES}[\![k, \alpha_i^x]\!]_x^i)\,.\end{aligned}$$

There are two cases to consider for $\mathrm{RES}[\![k, \alpha]\!][j_1, \ldots, j_m, j]$. First of all, assume that j is one of the parameters in k or α, say i_3. In this case, $\mathrm{RES}[\![k, \alpha]\!][j_1, \ldots, j_m, j]$ is a theorem iff $\mathrm{RES}[\![k, \alpha_{i_3}^x]\!][j_1, \ldots, j_m]$ is a theorem since all the other disjuncts of $\mathrm{RES}[\![k, \alpha]\!]$ will be inconsistent for this choice of x. However, by induction, $\mathrm{RES}[\![k, \alpha_j^x]\!][j_1, \ldots, j_m]$ is a theorem iff $(k \supset \alpha_j^x)[j_1, \ldots, j_m]$ is a theorem and this sentence is just $(k \supset \alpha)[j_1, \ldots, j_m, j]$. The other case to consider is when j is not a parameter of α or k. In this case, $\mathrm{RES}[\![k, \alpha]\!][j_1, \ldots, j_m, j]$ is a theorem iff $\mathrm{RES}[\![k, \alpha_i^x]\!]_x^i[j_1, \ldots, j_m, j]$ is a theorem since all other disjuncts of $\mathrm{RES}[\![k, \alpha]\!]$ will be inconsistent for this choice of x. Now let $*$ be the bijection over parameters that swaps i and j and leaves all other parameters unchanged. Then, by Corollary 3.4 $\mathrm{RES}[\![k, \alpha_i^x]\!]_x^i[j_1, \ldots, j_m, j]$ is a theorem iff $\mathrm{RES}[\![k, \alpha_i^x]\!]_x^i[j_1^*, \ldots, j_m^*, i]$ is a theorem. This sentence is just $\mathrm{RES}[\![k, \alpha_i^x]\!][j_1^*, \ldots, j_m^*]$ and, by induction, is a theorem exactly when $(k \supset \alpha_i^x)[j_1^*, \ldots, j_m^*]$ is. But this sentence is $(k \supset \alpha)[j_1^*, \ldots, j_m^*, i]$, which by Corollary 3.4 again, is a theorem iff $(k \supset \alpha)[j_1, \ldots, j_m, j]$ is.

We can now prove that RES[[k, α]] captures within a first-order formula what it means for a KB whose knowledge is represented by k to know that α is true.

Lemma 3.6. *For any sentence k of $\mathscr{L}$, any formula α of $\mathscr{L}$ and any parameters $j_1, \ldots, j_m$, the sentence* $(\boldsymbol{K}\alpha \equiv \text{RES}[[k, \alpha]])[j_1, \ldots, m_m]$ *is true on* $\mathscr{R}[k]$.

Proof. Suppose RES[[k, α]][$j_1, \ldots, j_m$] is false on $\mathscr{R}[k]$. It cannot, therefore, be valid and so, by Lemma 3.5, neither can $(k \supset \alpha)[j_1, \ldots, j_m]$. So, there must be a world structure on which k is true and $\alpha[j_1, \ldots, j_m]$ is false. Thus, $\boldsymbol{K}\alpha[j_1, \ldots, j_m]$ is false on $\mathscr{R}[k]$. Conversely, assume that RES[[k, α]][$j_1, \ldots, j_m$] is true on $\mathscr{R}[k]$. Since this sentence is pure and contains no $\boldsymbol{K}$-operators, it must be valid. So, again by Lemma 3.5, $(k \supset \alpha)[j_1, \ldots, j_m]$ must be valid also and, therefore, $\boldsymbol{K}\alpha[j_1, \ldots, j_m]$ must be true on $\mathscr{R}[k]$. To summarize, RES[[k, α]][$j_1, \ldots, j_m$] is true on $\mathscr{R}[k]$ iff $\boldsymbol{K}\alpha[j_1, \ldots, j_m]$ is.

Lemma 3.6 completes the proof of the Representation Theorem.

3.2.3. *Remarks*

The proof of the Representation Theorem clearly depends on being able to finitely characterize the set of entities known to have some property whenever the knowledge of the KB is finitely represented. To appreciate the non-triviality of this property, it is worth examining a slight modification to $\mathscr{L}$ (called $\mathscr{L}'$) for which the property no longer holds and, consequently, for which the Representation Theorem is false.

There are, of course, first-order dialects other than $\mathscr{L}$ that *do* satisfy the Representation Theorem. In fact, a dialect without any parameters at all will do the trick but in a very trivial way. Without standard names, there is no way to pick out any specific individual. For example, telling a KB that $\phi(c)$ is true, where c is a 0-ary function symbol, is like telling it $\phi(f(g(c, d)))$: it knows that some individual is a ϕ, but not which. In fact, it can be shown that the only time a KB will have a *known* ϕ is when it knows that everything is a ϕ. So the sentence

$$\exists x \boldsymbol{K}\phi(x) \supset \boldsymbol{K}\, \forall x \phi(x)$$

will end up being a theorem. What this means, then, is that RES[[k, α]] (where α has a single free variable) can be defined as $(x = x)$ or $(x \neq x)$, depending on k. Because the set of known instances of formulas is either everything or nothing, it can be characterized easily. This is enough to satisfy the Representation Theorem.[26]

Consider, on the other hand, a language $\mathscr{L}'$, like $\mathscr{L}$, except that it has a single parameter *1* and a special one-place *successor* function ′ (written in postfix

notation), that can be applied to any term. Instead of having terms like

$$1, 2, 3, \ldots,$$

there are terms

$$1, 1', 1'', \ldots$$

which play the same role in the language. We will call these terms *pseudo-parameters.* If the successor function was only used for pseudo-parameters, $\mathscr{L}'$ would just be a notational variant of $\mathscr{L}$. What makes the difference is that the successor function can be applied to *any* term including variables.

We will leave the semantics and proof theory of $\mathscr{L}'$ undefined since most of it is inherited from $\mathscr{L}$. One difference is that the Axiom of Equality can now be stated directly (without using a schema) as

$$\forall x((x = x) \wedge (1 \neq x') \wedge \forall y[(x' = y') \equiv (x = y)]).$$

Sentences of the form

$$(t_1 = t_1) \wedge (t_1 \neq t_2),$$

where t_1 and t_2 are distinct pseudo-parameters, are all implied by the new axiom. Moreover, if Universal Generalization remains unchanged except for dealing with pseudo-parameters instead of parameters, the converse is also true.

The language $\mathscr{KL}'$ is the modification of $\mathscr{KL}$ that has the successor function. The important point is that this function has to be special in $\mathscr{KL}'$ the same way that equality and parameters are special. In particular, a sentence can still be pure even if it contains an occurrence of the successor function symbol. Similarly, the Axiom of Specialization is not restricted in the placement of a $'$ symbol within the scope of a $\boldsymbol{K}$, as with other function symbols. What this amounts to is that just as a KB is assumed to know the truth value of $(t_1 = t_2)$ whenever it knows the referent of t_1 and t_2, it must know the referent of t' whenever it knows the referent of t. Another way of looking at this is to think

[26]As discussed in [7], this argument has more general implications for logics of knowledge and belief. It appears, for example, that if we assume that what an agent knows can be represented symbolically in a language (or, equivalently, that what he knows is the result of being told sentences in a language), then either there has to be rigid designators in that language or the kind of 'wh-knowledge' (or *de re* knowledge) he will have will be limited as above. In systems without rigid designators that do not have the above sentence as a theorem, there is a (perhaps implicit) assumption that wh-knowledge comes in some other form.

of parameters as *rigid designators* in that they always stand for the same things no matter how the other function and predicate symbols are interpreted (relative to an s and v). Similarly, equality is a *rigid predicate* in that it always stands for the identity relation. So the successor function is a *rigid function* since it too always stands for the same function. The key property about these symbols is that because they are effectively *logical* symbols, the KB must have complete knowledge of them. If it did not, pseudo-parameters could not be used to play the role of parameters.

However intuitively appealing this dialect may be, it cannot be used as a representation language since it is too expressive to satisfy the Representation Theorem. To see this, consider, to the contrary, the knowledge represented by

$$k = (\phi(1) \wedge \forall x[\phi(x) \supset \phi(x')]) .$$

What are the known ϕ in this case? Certainly 1 is one. Moreover, since the KB knows that $\phi(1')$ is true, $1'$ is a known ϕ also. In fact, every 'odd' pseudo-parameter is a known ϕ. The question then is how to represent in $\mathscr{L}'$ the result of

$$\text{TELL}[\![\mathscr{R}[k], \exists x[\psi(x) \wedge \boldsymbol{K}\phi(x))]]\!] .$$

What should RES$[\![k, \phi(x)]\!]$ be? There is a real problem here since instances of this formula must be either valid or unsatisfiable. Specifically, we cannot use something like number theory to characterize the odd pseudo-parameters since that would presumably involve using non-logical predicate or function symbols, thereby making the interpretation of the formula world dependent. It seems that the best we can do is use the infinite formula

$$(x = 1) \vee (x = 1') \vee (x = 1''') \vee \cdots .$$

The conclusion is that $\mathscr{L}'$ is too powerful a language since it allows a finitely represented KB to have a set of known ϕ that cannot be finitely characterized.

This does not mean that we cannot use $\mathscr{L}$ to provide a KB with a (finitely axiomatized) number theory of some sort. This theory may very well use a function symbol ' to mean the successor function over numbers. Any such KB would obviously know all the implications of the theory.[27] In this situation, however, the successor function would not be rigid. The KB need not know how the numerals (as represented by repeated applications of the successor function to some initial parameter or constant) line up with the parameters (even if the theory stipulates that everything is a number). For instance, given the knowledge represented by k above (in addition to some number theory),

[27]It could not, however, know all the *truths* of number theory, as Gödel has shown.

the parameter l would be the *only* known ϕ even though the KB would know that there were infinitely many. Number theory would play a role no more special than a theory about cities and states.

The implication of the above argument is that even a small part of number theory (in particular, a theory of the successor function) cannot be incorporated into the *logic* of $\mathscr{L}$ without losing the Representation Theorem. In other words, knowledge resulting from being told sentences in $\mathscr{L}'$ augmented by the $\boldsymbol{K}$ operator *cannot* be represented in $\mathscr{L}'$ itself. This impossibility is not a result of computational intractability or even of undecidability; there is no oracle that could be defined that would help us represent the knowledge. It is purely a property of what $\mathscr{L}'$ can and cannot express. What all this suggests is that the language $\mathscr{L}$ is very close to the limit of expressibility that a first order dialect can have and still satisfy the Representation Theorem. The characterization of what changes are in fact possible remains an open problem.

4. Extensions and Applications

Having investigated some of the technical issues underlying a knowledge level view of a KB, we will now examine some possible extensions to the framework. Specifically, new knowledge level capabilities will be introduced in terms of extensions to the TELL and ASK operations. In the first subsection, we consider an alternative form of ASK that handles wh-questions. Next, we consider a new ASK (and TELL) that provides a knowledge level view of non-monotonic reasoning. Finally, we briefly consider a TELL that allows definitional mechanisms to be incorporated into the framework. Although all these preliminary extensions are in need of further research, they should at least suggest how the framework can be applied to some of the current research problems in knowledge representation.

4.1. Wh-questions and numeric questions

One of the original motivations behind the use of $\mathscr{KL}$ as a query language for a first-order KB was for those situations where wh-questions were problematical. We can think of a wh-question as a formula with free variables and the answer(s) as being the parameter substitutions that make the formula true. For example, the formula

MajorCity(city, state)

asks what the major cities and their states are. The problem with this question is that a KB may not have a list of major cities and states, but only general pieces of information about them. In this case, it is not even clear what the *form* of an answer should be.

A solution to this problem proposed in [13] is to consider wh-questions as

coming in two forms: the first form asks for *known* instances of the formula; the second form asks for *potential* instances of the formula.[28] Obviously, the known instances are a subset of the potential instances. But more importantly, as far as the KB is concerned, the *actual* instances fall somewhere between the known and the potential ones. This is a consequence of the fact that $\boldsymbol{K}(\boldsymbol{K}\alpha \supset \alpha)$ and $\boldsymbol{K}(\alpha \supset -\boldsymbol{K}-\alpha)$ are both theorems of $\mathscr{KL}$. So, as observed by Lipski, the known and potential instances of a formula place lower and upper bounds respectively on the set of actual instances. Moreover, unlike the actual instances of a formula, a KB (that has complete knowledge about itself) always knows what the known and potential instances are.

One thing to notice about this form of answers to wh-questions is that if formulas are taken from $\mathscr{KL}$, it is sufficient to deal with known instances. Instead of asking for known and potential instances of α, we instead simply ask for the known instances of $\boldsymbol{K}\alpha$ and $-\boldsymbol{K}-\alpha$ respectively. This suggests that a wh-question facility can be defined by

$$\text{Wh-Ask}[\![k, \alpha]\!] = \{\langle i_1, \ldots, i_n\rangle \mid \boldsymbol{K}\alpha[i_1, \ldots, i_n] \text{ is true on } k\}.$$

The question itself determines whether known or potential instances will be retrieved, with known instances as the default (for example, when formulas of $\mathscr{L}$ are used).

Before dealing with the issue of what to do when the set of substitutions is infinite (and therefore, cannot be 'returned' by Wh-Ask), it is worth considering the form of an answer when the set is finite. Typically, a user will not be interested so much in a set of parameters since these only represent equivalence classes and have no descriptive import. For example, if the answer to the question

MajorCity(city, California)

is the set $\{4, 7, 9\}$, not much is learned except that there are three cities. What is also required is way of moving from a vacuous designator to coreferential terms that have descriptive (or at least connotative) import. So the second facility that must be provided is a function

$$\text{DESCRIBE}[\![k, i]\!] = \{t \mid t \text{ is not a parameter and } \boldsymbol{K}(t = i) \text{ is true on } k\}.$$

Given a parameter, this function returns the set of all terms that are known to

[28]By a 'potential' instance, we mean a substitution of the free variables which is not known to be a negative instance of the formula.

be coreferential. The trouble, of course, is that there may be infinitely many such terms (even when the KB is representable).

There are a few ways of dealing with this. Because a KB can only mention finitely many function symbols, the terms have to use the same symbols repeatedly. The easiest way to define a finite DESCRIBE is to consider only primitive terms (that is, those with exactly one function symbol). For example, the parameter *4* above might be described by

$$\{\text{SanFrancisco}, \text{FavoriteCity}(3)\}.$$

Similarly, *3* might be described by

$$\{\text{FirstChild}(1, 6), \text{BestFriend}(5), \text{OldestBrother}(2)\}.$$

A simple brute force way of implementing this version of DESCRIBE would be to loop through all the function symbols used in the KB and, for each one, consider the application of the function to each tuple (of the arity of the function symbol) consisting of parameters mentioned in the KB (which are also finite in number), returning those that are known to be equal to the original parameter argument. Another way of dealing with the problem would be to return only those terms that use a function symbol fewer than some number of times (passed as an argument, perhaps). An ideal solution might be to devise a *grammar* for describing the set of terms to be returned. For example, the set

$$\{6, f(6), f(f(6)), \ldots\}$$

can be described by the grammar

$$\langle \text{solution} \rangle ::= 6 \mid f(\langle \text{solution} \rangle).$$

It is not clear how powerful this kind of grammar would have to be.

Returning now to Wh-Ask itself, we are faced with the problem that the set of substitutions itself need not be finite (an obvious example being the answer to the question $x \neq y$). However, in this case, there is a very definite way to characterize the infinite set of parameter tuples. Specifically, for any KB represented by k, we can define

$$\text{Wh-Ask}[\![\mathscr{R}[k], \alpha]\!] = |\boldsymbol{K}\alpha|_k.$$

What this means is that the result (after simplifications) of Wh-Ask will be a formula like

$$(x = 2 \wedge y = 8) \vee (x = 4 \wedge y \neq 7).$$

Determining actual instances given a formula of this kind is a completely straightforward process.

Before leaving the topic of extensions to ASK, it is worth considering a different variety of question: those dealing with the size of sets. In the same way a first-order KB is taken to have knowledge about what it does and does not know (without containing any explicit sentences to this effect), it makes sense to assume that a KB implicitly knows the size of certain sets. For example, if a KB knows that

$$\exists x \exists y((x \neq y) \wedge \forall z[\phi(z) \equiv (z = x \vee z = y)]),$$

then the set of known ϕ is empty and everything is in the set of potential ϕ so the two wh-questions are uninformative. However, the KB does know something very specific about the ϕ, namely, that there are exactly two of them. What would be convenient in this case is the ability to ask the KB how many instances of α there are (allowing *unknown* and *infinitely many* as answers). This does not seem to presume any special number theoretic abilities on the part of the KB other than the ability to count. In fact, we can *already* ask, for any given n, if there are (exactly, at most, at least) n instances by using existential quantifiers and inequalities. One can also imagine a slightly more general question which is to name the greatest known lower bound and the least known upper bound on the size of the set of instances. Even with no information at all, a KB could return $\langle 0, \infty \rangle$ as the interval and subsequently sharpen these bounds as knowledge is acquired. In fact, the knowledge the KB has about the size of the set can be arbitrarily vague. For example, a KB that knows the negation of the above sentence only knows that the number of ϕ there are is not two. So it might be worth developing a range of questions to find out exactly what a KB knows about the number of instances of some formula. On a different axis, when dealing with formulas with several free variables, we will want to consider questions that talk about mappings between the variables. For example, answers to questions like

How many y's per x satisfy $\alpha[x, y]$?

are much more informative than the size of the set of the tuples. Again, the range of questions to consider here is completely open ended.

4.2. Default reasoning

There is an interesting parallel between the way RES$[\![k, \alpha]\!]$ works and the circumscription mechanism of [14]. The parallel is that while circumscription is a formal method for conjecturing that the only entities satisfying a predicate are those that must satisfy it (according to some theory), RES$[\![k, \alpha]\!]$ is a method

of getting hold of those entities that are known to satisfy it. If we let $\vdash_\phi$ be the provability relation when the predicate ϕ is circumscribed, we get, for example,

$$(\phi(1) \wedge \phi(2)) \vdash_\phi \forall x(\phi(x) \equiv (x = 1 \vee x = 2)).$$

In other words, if *1* and *2* are ϕ, then (if the only ϕ are those that must exist), *1* and *2* are the only ϕ. In our case, we can start by defining a provability relation $:\!\vdash$ by

$$k :\!\vdash \alpha \quad \text{iff} \quad \vdash (k \supset |\alpha|_k)$$
$$\text{where } k \in \mathscr{L} \text{ and } \alpha \in \mathscr{KL}.$$

One reading of this non-monotonic operator is

> If k represents everything that is known,
> then α is known to be true.

So here, for example, we have that

$$(\phi(1) \wedge \phi(2)) :\!\vdash \forall x(\boldsymbol{K}\phi(x) \equiv (x = 1 \wedge x = 2)).$$

This says that if it is known that *1* and *2* are ϕ, then if nothing else is known, *1* and *2* are the only *known* ϕ. Note that unlike circumscription, there is no conjecture here; there may very well be other instances of ϕ, but *1* and *2* are indeed the only known ones. However, we could get exactly the same result as circumscription by conjecturing that every ϕ is known, that is by adding the *closed world assumption* [15] for ϕ.[29] In this case, the ϕ would be identified with the known ϕ and the KB would then believe that *1* and *2* are the only ϕ. In other words, $\mathscr{KL}$ allows us to divide circumscription into two components: first, identify the known instances of a predicate, then conjecture that every instance is known.

This is not to say that there is an *exact* correspondence between circumscription and RES$[\![k, \alpha]\!]$ (with ϕ replaced by $\boldsymbol{K}\phi$). It appears that neither formal system can simulate the other. For example, with a theory containing only $\exists x\phi(x)$, circumscription would conclude that there was a *single* ϕ, while there are no known ϕ at all. On the other hand, the similarity suggests that $\mathscr{KL}$ might be a reasonable place to consider formulating yet another account of non-monotonic reasoning [16]. In particular, reasoning by default [17] seems to consist of making assumptions *in the absence of knowledge to the contrary*. It is this auto-epistemic aspect that is not addressed in other formalizations and that makes the language $\mathscr{KL}$ especially relevant.

[29] An important property of $\mathscr{KL}$ is that this closed world assumption can be expressed as a sentence: what we want to TELL the KB is that $\forall x(\phi(x) \supset \boldsymbol{K}\phi(x))$.

The idea behind the treatment of defaults suggested in [7] is the following. Consider two query operators: ASK, as before, and ASK* which answers taking defaults into account. For example, given a KB whose knowledge is represented by Bird(*1*), we would have that

$$\text{ASK}[\![\mathscr{R}[\text{Bird}(1)], \text{Fly}(1)]\!] = \text{unknown},$$

but

$$\text{ASK}*[\![\mathscr{R}[\text{Bird}(1)], \text{Fly}(1)]\!] = \text{yes}.$$

In other words, ASK* is willing to assume by default that the bird in question flies.

This effect can be defined precisely in terms of an ASK* that behaves exactly like ASK except with respect to a KB that has some additional beliefs about birds. In other words, for some δ, ASK* here could be characterized[30] by

$$\text{ASK}*[\![k, \alpha]\!] = \text{ASK}[\![\text{TELL}[\![k, \delta]\!], \alpha]\!].$$

The δ should state that 'by default' birds fly. One way to paraphrase this, given $\mathscr{KL}$, is to state that all birds fly except those known not to fly:

$$\delta = \forall x([\text{Bird}(x) \wedge \neg\boldsymbol{K} \neg\text{Fly}(x)] \supset \text{Fly}(x)).$$

An alternative formulation might be to say that all *known* birds fly except those known not to fly (that is, the first predicate in δ would be $\boldsymbol{K}\text{Bird}(x)$). One difference between the two is that in the original formulation only, with no information at all, a KB would assume that all birds fly. Other differences are discussed in [7].

A sentence of this type might be called a *default*: a statement that certain individuals have a property provided that they are not known not to have the property. So like the formalism of [18] but unlike that of [19], defaults are sentences of a language, that is, potential objects of belief which have a very natural proof theoretic *and* semantic characterization by virtue of the Representation Theorem. Moreover, unlike both of these approaches, ours does not require defining a logic with new inference rules[31] or sentential operators. In fact, the sentential operator $\boldsymbol{M}$ of [18] seems to be related to $\boldsymbol{K}$. Their interpretation of $\boldsymbol{M}\alpha$ is that α is *consistent* with what is believed. However, it is not clear how important it is to interpret the symbol in terms of what can or cannot be *derived* in a formal system. It seems more appropriate to interpret it (epistemically) as saying that α is not *known* to be false, that is, as $\neg\boldsymbol{K}\neg\alpha$.

There is another approach to defaults we might consider, which is to leave ASK unchanged but provide a new function TELL* that behaves like TELL

[30]It is important to realize that this definition of ASK* only states what the behavior of defaults must be and not how they could be *implemented* at the symbol level.

[31]See [12] for arguments about why non-monotonic belief revision should not be construed as following a logic with non-monotonic inference rules.

but taking defaults into account. The difference between the two methods is that ASK* answers questions by first making some assumptions while leaving the KB unchanged; TELL*, on the other hand, would actually change the KB allowing the default to become a regular belief. In other words, ASK* says that defaults are assumptions that can be made to answer a certain form of question, while TELL* says that defaults are beliefs that are acquired by virtue of what is left unsaid in a certain form of assertion. This distinction is difficult if not impossible to capture within existing formalizations of default reasoning but is handled quite naturally in our functional framework.

Obviously, a general definition of ASK* (or TELL*) should not depend on the default properties of birds specifically. A more reasonable definition that depends only on the query and the current KB is presented in [7]. Essentially, the idea is to separate a default into two pieces. The first involves knowledge about typical birds, for example, that they fly, have two wings, and so on. A new predicate forming operator ∇ (read as '*typical*') is introduced so that if "Bird" is a one place predicate, then so is '∇Bird'.[32] So, for example, a KB may know that

$$\forall x(\nabla \text{Bird}(x) \supset \text{Fly}(x)) .$$

without having any idea of which birds are typical (although it will realize that any bird that does not fly is atypical). This is not a belief about all birds in general, just the typical ones. But the crucial points is that this is *not* default knowledge: the belief here is that typical birds fly—not that typical birds *typically* fly. In fact, we may want to say something stronger than this, namely, that flying is not only not a default property of typical birds, it is an *essential* property. The impact of this distinction will be taken up again below.

The second part of a default says which birds are typical and is the sentence that will be assumed by ASK*:

$$\forall x([\text{Bird}(x) \wedge \text{—}\boldsymbol{K}\text{—}\nabla \text{Bird}(x)] \supset \nabla \text{Bird}(x)) .$$

In other words, the assumption is that any bird not known to be atypical is typical. Although this sentence uses the predicate 'Bird', it would have exactly the same form for any other predicate. In fact, δ can be defined (almost) as the conjunction of sentences of this form over all predicates used in the KB (except those preceded by a ∇).[33] Returning to our above example, the end result is

[32]Obviously we want ∇ to apply to n place predicates for any n. Moreover, because a bird can be typical with respect to number of legs but atypical with respect to flying ability, there will be a family of predicates formed by ∇, given a predicate and an index.

[33]The simple conjunctive definition has to be modified because of the complications that arise when trying to apply multiple competing defaults simultaneously. These issues are discussed in [20] and [7].

that while the original KB will not know if the bird flies, it will believe that if it is typical, then it flies. Moreover, the augmented KB that is used for ASK* will also assume that it is typical (since there is no knowledge of it being atypical) and, therefore, that it flies.

To summarize, the notion of default that seems the best suited to $\mathscr{KL}$ is a very simple one where certain instances of predicates are assumed to be typical. All of the 'content' of the default is put into knowledge about the properties of typical instances of the predicates. While much remains to be done on the semantics of the ∇-operator, there is already evidence from other sources that it is a useful abstraction [21].

4.3. Definitional mechanisms

In terms of system design, the main reason for distinguishing between the knowledge level and the symbol level is to allow the functionality of a system to be treated independently of its symbolic implementation. In particular, it allows us to consider new operations on a KB (that can be explained in terms of existing ones) without necessarily committing ourselves to any particular implementation style.

Consider an operation that allows new non-logical terms to be *defined* in terms of existing ones.[34] For example, we may decide to start using a predicate 'CapitalCity' and want it to apply only to cities that are state capitals. One way of informing a KB of this is to assert (using TELL) something like

$$\forall x(\text{CapitalCity}(x) \equiv \text{City}(x) \wedge \exists y(\text{State}(y) \wedge \text{Capital}(y) = x)) .$$

However, even though this assertion gets the KB to believe the right things about capitals, it does not tell it that this is how the term is being defined. To see this, suppose we just wanted to tell the KB that (it so happened that) someone called Joe liked every city except the capitals. We could then justifiably assert

$$\forall x(\text{CapitalCity}(x) \equiv \text{City}(x) \wedge \text{—Like}(\text{Joe}, x)) .$$

Notice that the form of these two sentences is very similar; if either one is supposed to be a *definition* of 'CapitalCity', then they should both be definitions. But we certainly do not want to *define* capital cities as those that Joe does not like. Nor, for that matter, do we want to say that every capital city just so happens to be a capital of some state, as if it were completely by accident or even possible that some were not. Indeed, the trouble with trying to identify definitions in terms of the logical form of certain assertions is that not only are some universally quantified biconditionals not definitional, there may

[34] See [4] for arguments against the prevalent position that definitions are only relevant in formal domains (such as mathematics).

very well be definitions of terms that are not universally quantified biconditionals.

It should be obvious that our TELL operation does not capture the difference in intent between the two sentences. Its purpose is to allow a KB to be told true sentences without regard to whether the truth is based on a property of the world or on the definition of a term. As it stands, there is no way to *define* a term like 'CapitalCity' so as to cause sentences like the first to be believed. To the extent that this kind of functionality is deemed useful in the design of knowledge-based systems (see [22], for example), a new operation is required.

The idea here is to have an operation

$$\text{DEFINE}[\![k, \text{term}, \text{definition}]\!]$$

that takes a KB, a predicate (or function) symbol and a definition for this predicate and has as value a KB that knows how the term is defined. The actual form of a definition is not really relevant here, though what we have in mind is a small number of *predicate forming operators* that allow complex predicates to be formed out of simpler ones. For example, a predicate forming operator Qua could be used to define capitals as those cities that play the role of state capital for some state:

$$\text{DEFINE}[\![k, \text{CapitalCity}, (\text{Qua City Capital State})]\!] .$$

The only major question to answer here is what the definition of the DEFINE operation should be. Perhaps the simplest solution is to treat a definition exactly as before, that is, as synonymous with an assertion of some sentence, a *meaning postulate* associated with the term forming operator. For example, we could understand an operation like

$$\text{DEFINE}[\![k, \phi, (\text{Qua } \psi\ \xi\ \theta)]\!]$$

as if it were

$$\text{TELL}[\![k, \forall x(\phi(x) \equiv \psi(x) \wedge \exists y[\theta(y) \wedge \xi(y) = x])]\!] .$$

This form of 'macro'-definition performs exactly the same assertion as an explicit TELL except that we have separated the actual term formation from the meaning postulate.

Even though we can explain a DEFINE in terms of a TELL, there are good reasons for keeping the two separate. First of all, the two operations are different in intent and thus have different standards of adequacy. For example, the adequacy of the TELL operation is measured in terms of its ability to form very weak assertions. Incompleteness is not an issue with term formation, however. As argued in [22], what is likely to count in a definitional facility is

the ability to generalize and specialize existing terms and to aggregate collections of terms into larger complex wholes.

Moreover, the distinction between the two operations encourages us to consider implementation methods for definitions that are not based (in the obvious way) on the implementation used for the TELL operation. For example, under the (very plausible) assumption that it will be convenient to move quickly from a term to its definition, we might consider using an inheritance network data structure to keep track of relationships among the various terms being defined. In this sense, a definition allows us to inform the symbol level about what sentences of a certain form are important enough to be treated specially, something that would not be possible if only assertions were allowed.

Finally, even without considering any additional operations, we can present an interface to a KB that appears to extend beyond the first-order capabilities. For example, we might allow

$$\text{DEFINE}[\![k, \phi, (\text{TransitiveClosure } \psi)]\!]$$

to mean

$$\text{TELL}[\![k, \forall x \forall y(\phi(x, y) \equiv (x = y) \vee \exists z[\psi(x, z) \wedge \phi(z, y)])]\!] .$$

This would, for example, allow us to define 'ancestor' directly as the transitive closure of 'parent' without having to characterize a transitive closure relation over predicates using a second order theory. As it turns out, transitive closures also happen to be predicates that are best implemented at the symbol level in terms of special-purpose inference techniques.

If we consider an extension to our framework that allows new *query* operations, the difference between defining a term and asserting a sentence becomes more substantial. For example, if we can ask if one term analytically subsumes[35] another, then we can obviously notice the difference between defining a term in a certain way and asserting some corresponding sentence (meaning postulate). The same is true if we can access the definition of a term directly. This kind of capability allows us to examine how a predicate was defined independently of what is known about (instances of the) predicate. In a sense, our functional view of a KB is enlarged to reflect a distinction between *essential* properties of instances of a term (arising by virtue of the way the term was defined) and *factual* ones (arising by virtue of what is true in the world).

The issue here, of course, is how to define these extended operations. It is no longer sufficient, for example, to explain DEFINE in terms of a TELL if we have to be able to calculate subsumption relationships. The simplest solution

[35]A predicate analytically subsumes another if, by virtue of how the predicates are defined, every instance of the latter predicate must also be an instance of the former.

might be to divide a KB into two parts, the first as before, and the second containing the effect of the definitions. The regular ASK operation would use both parts of the KB while analytic questions would only consider the second part. This second component could be a set of world structures, or (syntactic) definitions of some sort, or even meaning postulates. A more radical possibility is to make the definitional component incorporate special relations over predicates that are part of the semantics of the language itself. This is essentially what is done in [23], where a lattice structure over predicates (where the partial ordering is that of conceptual containment) is part of the model theory of a first-order language.

Another possible extension to our framework is to allow for the definition of predicates which cannot be characterized in terms of necessary or sufficient conditions. For such terms (like those standing for *natural kinds*) any 'definition' is best thought of in terms of *prototypical* rather than *essential* properties. These are properties that instances of the term might be *expected* to have. If we take these prototypical properties to be part of how the predicate is defined, a simple approach suggests itself. Basically, we can use the default mechanism discussed above and consider the definition of any predicate ϕ to include the definition of $\nabla\phi$. So, to make *flying* be a prototypical property of birds, we make it be an essential property of *typical* birds. In other words, the prototypical properties of 'Bird' are just the essential properties of '∇Bird'. Overall, then, instances of a predicate would be characterized in terms of factual, essential, and prototypical properties (leaving open, for the moment, the epistemological status of the factual properties of instances of '∇Bird').

This simple expedient allows us to consider definitions of terms that include both essential and prototypical features. In other words, we can support the kind of 'definition' found in frame-based representation languages like UNITS [24] or KRL [25] (modulo syntactic sugar, of course). But most importantly, because we have separated the knowledge level from the symbol level, we can *implement* these definitions in any convenient way that preserves the semantics of the operations. Specifically, we are free to use the same methods found useful in the frame-based languages (such as inheritance networks), rather than more general theorem proving techniques better suited to TELL and ASK.

To summarize, if we extend the framework to incorporate definitional facilities and especially if we include default capabilities, we can provide a framework that has aspects of both logical and object-oriented representation systems. As such, it offers an especially convenient formal starting point for the development of 'hybrid' systems like the kind discussed in [22, 23, 26].

5. Conclusion

In this paper, we have investigated some of the technical properties of a functional approach to knowledge representation where *what* a KB does for a

system is kept quite separate from *how* it represents what it knows. This, we argued, was in keeping with current programming principles (exemplified in the abstract data type methodology), as well as being compatible with Newell's notion of knowledge and symbol levels.

Our first concern was to develop a language for *interacting* with a KB, a first order logical language called $\mathscr{L}$. Because the usual role of logical languages in AI has been to store information (at the symbol level), our dialect had to have special features such as parameters and a non-standard treatment of equality. We illustrated the appropriateness of this dialect by showing that a slightly stronger one (called $\mathscr{L}'$) was too expressive to satisfy the Representation Theorem.

We then showed that using $\mathscr{L}$ as an assertion language can result in an incomplete KB that is best queried in a more powerful language. This led to a generalization of $\mathscr{L}$, called $\mathscr{KL}$, which allows reference to not only the domain, but to the KB's knowledge of the domain, its knowledge of its knowledge of the domain, and so on. We gave a formal semantics for $\mathscr{KL}$ independent of the one for $\mathscr{L}$, something we could not have done if $\mathscr{KL}$ had merely been a first-order theory (using syntactic encodings of sentences).

The important property of $\mathscr{KL}$ as an interaction language is that it allowed closed (or open) world assumptions and defaults, among other things, to be expressed as sentences of the language. Since TELL and ASK were defined in terms of $\mathscr{KL}$, a KB could be informed or queried about these sentences. The result is that, unlike other approaches, a KB could be made to believe closed world assumptions selectively like any other knowledge about the domain.

The major technical result of this research, the Representation Theorem, establishes that even with the extended expressive power provided by $\mathscr{KL}$, the knowledge of a KB is still representable at the symbol level using sentences of $\mathscr{L}$. In other words, the theorem shows the *correctness* of a first order symbolic realization of a KB. Again, had $\mathscr{KL}$ been specified as a first-order theory, no such proof would have been possible since the behavior of a KB would have been *defined* in terms of its first-order implementation.

In the final section, we argued for the robustness (and relevance) of our approach by showing how it could be applied to some current knowledge representation problems. The danger in a formal logical investigation like the one we have undertaken is that results may be regarded as 'merely' formal or mathematical, with no relevance to real knowledge-based systems. Our feeling is that the functional viewpoint does have *practical* advantages which we are currently attempting to exploit in a knowledge representation system called KRYPTON [27]. The two main features of KRYPTON are an interface based on TELL and ASK and a division of the functionality into definitional and assertional components along the lines discussed above.

Once knowledge level concerns have been separated from those at the symbol level, the kind of issues discussed here become very relevant to the

design of knowledge representation systems. The functional approach we have considered applies equally well to languages that are less expressive than $\mathcal{KL}$. Without advocating a specific position on the expressiveness *vs.* computational tractability trade-off, the framework offers a formal foundation for comparing and evaluating competing proposals by concentrating on the service provided by a KB independently of how this service is realized. Once the *competence* of the knowledge representation system has been established, its *performance* characteristics can be examined in proper perspective.

ACKNOWLEDGMENT

This research is based on my doctoral dissertation at the University of Toronto. I want to thank my thesis supervisor, Prof. John Mylopoulos, for his invaluable moral and technical support. I also wish to acknowledge my colleagues at the University of Toronto, BBN, and FLAIR for their many contributions to the ideas reported here and to thank in particular Ron Brachman, Phil Cohen, Joe Halpern, Gerhard Lakemeyer, Peter Patel-Schneider, Jim Schmolze, and an anonymous referee for their comments on an earlier draft of this manuscript. Financial support was gratefully received from the Department of Computer Science of the University of Toronto and the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

1. Newell, A., The knowledge level, *AI Magazine* **2**(2) (1981) 1–20.
2. Liskov, B. and Zilles, S., Programming with abstract data types, *SIGPLAN Notices* **9**(4) (1974).
3. Gallaire, H. and Minker, J. (Eds.), *Logic and Data Bases* (Plenum, New York, 1978).
4. Israel, D.J. and Brachman, R.J., Distinctions and confusions: A catalogue raisonne, *Proc. IJCAI-81*, Vancouver (1981) 452–459.
5. Mendelson, E., *Introduction to Mathematical Logic* (Van Nostrand–Reinhold, New York, 1964).
6. Levesque, H.J., The interaction with incomplete knowledge bases: A formal treatment, *Proc. IJCAI-81*, Vancouver (1981) 240–245.
7. Levesque, H.J., A formal treatment of incomplete knowledge bases, Tech. Rept. No. 3, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA, 1982.
8. Leblanc, H., On dispensing with things and worlds, in M.K. Munitz (ed.), *Logic and Ontology* (New York University Press, New York, 1973) 241–259.
9. Levesque, H.J., A formal treatment of incomplete knowledge bases, M. Brodie, J. Mylopoulos and J. Schmidt (Eds.), in: *Perspectives in Conceptual Modelling* (Springer, Berlin, 1984).
10. Hintikka, J., *Knowledge and Belief: An Introduction to the Logic of the Two Notions* (Cornell University Press, Ithaka, NY, 1962).
11. Moore, R., Reasoning about knowledge and action, Tech. Note 181, SRI International, Menlo Park, CA, 1980.
12. Israel, D.J., What's wrong with non-monotonic logic, *Proc. AAAI-80*, Stanford, CA, (1980) 99–101.
13. Lipski, W., On semantic issues connected with incomplete information data bases, *TODS* **4**(3) (1978).
14. McCarthy, J., Circumscription—A form of non-monotonic reasoning, *Artificial Intelligence* **13** (1980) 27–39.
15. Reiter, R., On closed world data bases, in: H. Gallaire and J. Minker (Eds.), *Logic and Data Bases* (Plenum, New York, 1978), 55–76.
16. Bobrow, D.G. (Ed.), *Artificial Intelligence* **13**, Special Issue on Non-Monotonic Reasoning, 1980.

17. Reiter, R., On reasoning by default, *Proc. Second TINLAP Conference*, Urbana, IL (1978) 210–218.
18. McDermott, D. and Doyle, J., Non-monotonic logic I, *Artificial Intelligence* **13** (1980) 41–72.
19. Reiter, R., A logic for default reasoning, *Artificial Intelligence* **13** (1980) 81–132.
20. Reiter, R. and Criscuolo, G., Some representational issues in default reasoning, Tech. Rept. 80–7, Dept. of Computer Science, University of British Columbia, Vancouver, 1980.
21. Cohen, B., Understanding natural kinds, Ph.D. Thesis, Dept. of Philosophy, Stanford University, Stanford, CA, 1982.
22. Brachman, R.J. and Levesque, H.J., Competence in knowledge representation, Proc. *AAAI-82*, Pittsburgh, PA (1982) 189–192.
23. Israel, D.J., On interpreting network formalisms, *Internat. J. Comput. Math.* **9**(1) (1983) 1–14.
24. Stefik, M., An examination of a frame-structured representation system, *Proc. IJCAI-79*, Tokyo (1979) 845–852.
25. Bobrow, D.G. and Winograd, T., An overview of KRL: A knowledge representation language, *Cognitive Sci.* **1**(1) (1977) 3–46.
26. Rich, C., Knowledge representation languages and predicate calculus: How to have your cake and eat it too, *Proc. AAAI-82*, Pittsburgh, PA (1982) 193–196.
27. Brachman, R.J., Fikes, R.E. and Levesque, H.J., KRYPTON: A functional approach to knowledge representation, *IEEE Comput.* **16**(10) (1983) 67–73.

Received March 1983