4 / Hector J. Levesque and Ronald J. Brachman
A Fundamental Tradeoff in Knowledge Representation and Reasoning (Revised Version)

If we assume Brian Smith's Knowledge Representation Hypothesis [Smith, Chapter 3], we can interpret the data structures in a Knowledge Representation system as if they were declarative sentences. In this paper, Levesque and Brachman argue that the proper role of such a system is to perform a class of inferences determined by the truth conditions of these sentences. Among other things, the system should be able to determine when the truth of one sentence is implicit in another. However, depending on the range of sentences that have to be dealt with, this task can be relatively trivial or completely unsolvable. Thus, as the title suggests, there is a tradeoff between the expressiveness and the tractability of a Knowledge Representation scheme. The paper illustrates this point by looking at a variety of formalisms (simple databases, logic programs, semantic networks, and frame systems) in terms of the range of first-order logic sentences they can express and the kind of reasoning that they require. One observation is that expressiveness often amounts to the ability to leave certain things unsaid, with full first-order logic at one extreme and databases (and "analogues") at the other. The main conclusion of the survey, however, is that neither expressiveness nor tractability by itself determine the value of a representation language (Allen's time representation [Chapter 30] is given as an example of a formalism that has quite consciously and very effectively traded expressive power for computational advantage). There is no single *best* language, it is argued, only more or less interesting positions on the tradeoff. The paper encourages the design of Knowledge Representation languages with this dimension in mind, contrary perhaps to the views in [Hayes, Chapter 28] and [Moore, Chapter 18] regarding the suitability of full first-order logic as *the* representation framework. It also presents some surprising results about the complexity of computing a kind of inference in simple frame representations.

Original version appeared as "A Fundamental Tradeoff in Knowledge Representation and Reasoning" (by Hector J. Levesque), *Proc. CSCSI-84*, London, Ontario, 1984, 141–152.

A Fundamental Tradeoff in Knowledge Representation and Reasoning (Revised Version[1])

Hector J. Levesque
Ronald J. Brachman

May, 1985

Abstract

A fundamental computational limit on automated reasoning and its effect on Knowledge Representation is examined. Basically, the problem is that it can be more difficult to reason correctly with one representational language than with another and, moreover, that this difficulty increases dramatically as the expressive power of the language increases. This leads to a tradeoff between the expressiveness of a representational language and its computational tractability. Here we show that this tradeoff can be seen to underlie the differences among a number of existing representational formalisms, in addition to motivating many of the current research issues in Knowledge Representation.

1 Introduction

This paper examines from a general point of view a basic computational limit on automated reasoning, and the effect that it has on Knowledge Representation (KR). The problem is essentially that it can be more difficult to reason correctly with one representational language than with another and, moreover, that this difficulty increases as the expressive power of the language increases. There is a tradeoff between the expressiveness of a representational language and its computational tractability. What we attempt to

[1]This is a revised version of "A Fundamental Tradeoff in Knowledge Representation and Reasoning," by Hector J. Levesque, which appeared in the *Proceedings of the CSCSI/SCEIO Conference 1984*, London Ontario, May, 1984. It includes substantial portions of two other conference papers: "The Tractability of Subsumption in Frame-Based Description Languages," by Ronald J. Brachman and Hector J. Levesque, which appeared in *Proceedings of AAAI-84*, Austin, Texas, August, 1984; and "What Makes a Knowledge Base Knowledgeable? A View of Databases from the Knowledge Level," by the same authors, which appeared in *Proceedings of the First International Workshop on Expert Database Systems*, Kiawah Island, South Carolina, October, 1984.

show is that this tradeoff underlies the differences among a number of representational formalisms (such as first-order logic, databases, semantic networks, frames) and motivates many current research issues in KR (such as the role of analogues, syntactic encodings, and defaults, as well as the systems of limited inference and hybrid reasoning).

To deal with a such a broad range of representational phenomena, we must, of necessity, take a considerably simplified and incomplete view of KR. In particular, we focus on its computational and logical aspects, more or less ignoring its history and relevance in the areas of psychology, linguistics, and philosophy. The area of KR is still very disconnected today and the role of logic remains quite controversial, despite what this paper may suggest. We do believe, however, that the tradeoff discussed here is fundamental. As long as we are dealing with computational systems that reason automatically (without any special intervention or advice) and correctly (once we define what *that* means), we will be able to locate where they stand on the tradeoff: they will either be limited in what knowledge they can represent or unlimited in the reasoning effort they might require.

Our computational focus will not lead us to investigate specific algorithms and data structures for KR and reasoning, however. What we discuss is something much stronger, namely whether or not algorithms of a certain kind can exist at all. So the analysis here is at the *Knowledge Level* [21] where we look at the content of what is represented (in terms of what it says about the world) and not the symbolic structures used to represent that knowledge. Indeed, we examine specific representation schemes in terms of what knowledge they can represent, rather than in terms of how they might actually represent it.

In the first section below, we discuss what a KR system is for and what it could mean to reason correctly. Next, we investigate how a KR service might be realized using theorem proving in first-order logic and the problem this raises. Following this, we present various representational formalisms and examine the special kinds of reasoning they suggest. We concentrate in particular on frame-based description languages, examining in some detail a simple language and a variant. In the case of this pair of languages, the kind of tradeoff we are talking about is made concrete, with a dramatic result. Finally, we draw some tentative general conclusions from this analysis.

2 The Role of Knowledge Representation

While it is generally agreed that KR plays an important role in (what have come to be called) knowledge-based systems, the exact nature of that role is often hard to define. In some cases, the KR subsystem does no more than manage a collection of data structures, providing, for example, suitable search facilities; in others, the KR subsystem is not really distinguished from the rest of the system at all and does just about everything: make decisions, prove theorems, solve problems, and so on. In this section, we discuss in very general terms the role of a KR subsystem within a knowledge-based system.

2.1 The Knowledge Representation Hypothesis

A good place to begin a discussion of KR as a whole is with what Brian Smith has called in [28] the *Knowledge Representation Hypothesis:*

> *Any mechanically embodied intelligent process will be comprised of structural ingredients that (a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and (b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.*

This hypothesis seems to underly much of the research in KR. In fact, we might think of *knowledge-based systems* as those that satisfy the hypothesis by design. Also, in some sense, it is only with respect to this hypothesis that KR research can be distinguished from any number of other areas involving symbolic structures such as database management, programming languages and data structures.

Granting this hypothesis, there are two major properties that the structures in a knowledge-based system have to satisfy. First of all, it must be possible to interpret them as *propositions* representing the overall knowledge of the system. Otherwise, the representation would not necessarily be of *knowledge* at all, but of something quite different, like numbers or circuits. Implicit in this constraint is that the structures have to be expressions in a language that has a *truth theory*. We should be able to point to one of them and say what the world would have to be like for it to be true. The structures themselves need not *look* like sentences—there are no syntactic requirements on them at all, other than perhaps finiteness—but we have to be able to understand them that way.

A second requirement of the hypothesis is perhaps more obvious. The symbolic structures within a knowledge-based system must play a *causal role* in the behaviour of that system, as opposed to, say, comments in a programming language. Moreover, the influence they have on the behaviour of the system should agree with our understanding of them as propositions representing knowledge. Not that the system has to be aware in any mysterious way of the interpretation of its structures and their connection to the world;[2] but for us to call it knowledge-based, *we* have to be able to understand its behaviour as if it believed these propositions, just as we understand the behaviour of a numerical program as if it appreciated the connection between bit patterns and abstract numerical quantities.

2.2 Knowledge Bases

To make the above discussion a bit less abstract, we can consider a very simple task and consider what a system facing this task would have to be like for us to call it knowledge-based. The amount of knowledge the system will be dealing with will, of course, be very small.

Suppose we want a system in PROLOG that is able to print the colours of various items One way to implement that system would be as follows:

```
printColour(snow) :-  !, write("It's white.").
printColour(grass) :- !, write("It's green.").
printColour(sky) :- !, write("It's yellow.").
printColour(X) :- write("Beats me.").
```

A slightly different organization that leads to the same overall behaviour is

```
printColour(X) :
    colour(X,Y), !, write("It's "), write(Y), write(".")
printColour(X) :- write("Beats me.").

colour(snow,white).
colour(grass,green)
colour(sky,yellow).
```

The second program is characterized by explicit structures representing the (minimal) knowledge[3] the system has about colours and is the kind of system that we are calling knowledge-based. In the first program, the association between the object (we understand as) referring to grass and the one referring to its colour is implicit in the structure of the

[2]Indeed, part of what philosophers have called the *formality condition* is that computation at some level has to be uninterpreted symbol manipulation.

[3]Notice that typical of how the term "knowledge" is used in AI, there is no requirement of *truth*. A system may be mistaken about the colour of the sky but still be knowledge-based. "Belief" would perhaps be a more appropriate term, although we follow the standard AI usage in this paper.

program. In the second, we have an explicit *knowledge base* (or KB) that we can understand as propositions relating the items to their colours. Moreover, this interpretation is justified in that these structures determine what the system does when asked to print the colour of a particular item.

One thing to notice about the example is that it is not the use of a certain programming language or data-structuring facility that makes a system knowledge-based. The fact that PROLOG happens to be understandable as a subset of first-order logic is largely irrelevant. We could probably read the first program "declaratively" and get sentences representing some kind of knowledge out of it; but these would be very strange ones dealing with writing strings and printing colours, not with the colours of objects.

2.3 The Knowledge Representation Subsystem

In terms of its overall goals, a knowledge-based system is not directly interested in what specific structures might exist in its KB. Rather, it is concerned about what the application domain is like, for example, what the colour of grass is. How that knowledge is represented and made available to the overall system is a secondary concern and one that we take to be the reponsibility of the KR subsystem. The role of a KR subsystem, then, is to manage a KB for a knowledge-based system and present a picture of the world based on what it has represented in the KB.

If, for simplicity, we restrict our attention to the yes-no questions about the world that a system might be interested in, what is involved here is being able to determine what the KB says regarding the truth of certain sentences. It is not whether the sentence itself is present in the KB that counts, but whether its truth is *implicit* in the KB. Stated differently, what a KR system has to be able to determine, given a sentence α, is the answer to the following question:

Assuming the world is such that what is believed is true, is α also true

We will let the notation KB $\models \alpha$ mean that α is implied (in this sense) by what is in the KB

One thing to notice about this view of a KR system is that the service it provides to a knowledge-based system depends only on the truth theory of the language of representation. Depending on the particular truth theory, determining if KB $\models \alpha$ might require not just simple retrieval capabilities, but also *inference* of some sort. This is not to say that the *only* service to be performed by a KR subsystem is question-answering. If we imagine the overall system existing over a period of time, then we will also want it to be able to augment the KB as it acquires new information about the world.[4] In other words, the responsibility of the KR system is to select appropriate *symbolic structures* to represent knowledge, and to select appropriate *reasoning mechanisms* both to answer questions and to assimilate new information, in accordance with the truth theory of the underlying representation language.

So our view of KR makes it depend only on the semantics of the representation language, unlike other possible accounts that might have it defined in terms of a set of formal symbol manipulation routines (e.g., a proof theory). This is in keeping with what we have called elsewhere a *functional* view of KR (see [15] and [5]), where the service performed by a KR system is defined separately from the techniques a system might use to realize that service.

3 The Logical Approach

To make a lot of the above more concrete, it is useful to look at an example of the kinds of knowledge that might be available in a given domain and how it might be represented in a KB. The language that will be used to represent knowledge is that of a standard first-order logic (FOL).

3.1 Using First-Order Logic

The first and most prevalent type of knowledge to consider representing is what might be called simple *facts* about the world, such as

- Joe is married to Sue
- Bill has a brother with no children.
- Henry's friends are Bill's cousins.

These might be complicated in any number of ways, for example, by including time parameters and certainty factors.

Simple observations such as these do not exhaust what might be known about the domain, however. We may also have knowledge about the *terminology* used in these observations, such as

[4] It is this management of a KB over time that makes a KR subsystem much more than just the implementation of a static deductive calculus.

- *Ancestor* is the transitive closure of *parent*
- *Brother* is *sibling* restricted to males
- *Favourite-cousin* is a special type of *cousin*

These could be called definitions except for the fact that necessary and sufficient conditions might not always be available (as in the last example above). In this sense, they are much more like standard dictionary entries.

The above two example sets concentrate on what might be called *declarative* knowledge about the world. We might also have to deal with *procedural* knowledge that focuses not on the individuals and their interrelationships, but on *advice* for reasoning about these. For example, we might know that

- To find the father of someone, it is better to search for a parent and then check if he is male, than to check each male to see if he is a parent.
- To see if x is an ancestor of y, it is better to search up from y than down from x.

One way to think of this last type of knowledge is not necessarily as advice to a reasoner, but as declarative knowledge that deals implicitly with the combinatorics of the domain as a whole.

This is how the above knowledge might be represented in FOL:

1. The first thing to do is to "translate" the simple facts into s ıces of FOL. This would lead to sentences like

   $$\forall x\,\mathrm{Friend}(\mathrm{henry}, x) \equiv \mathrm{Cousin}(\mathrm{bill}, x)$$

2. To deal with terminology in FOL, the easiest way is to "extensionalize" it, that is, to pretend that it is a simple observation about the domain. For example, the **brother** statement above would become[5]

   $$\forall x \forall y\,\mathrm{Brother}(x, y) \equiv (\mathrm{Sibling}(x, y) \wedge \mathrm{Male}(y)).$$

3. Typically, the procedural advice would not be represented explicitly at all in an FOL KB, but would show up in the *form* of (1) and (2) above. Another alternative would be to use extra-logical annotations like the kind used in PROLOG or those described in [19].

[5]This is a little misleading since it will make the *brother* sentence appear to be no different in kind from the one about Henry's friends, though we surely do not want to say that Henry's friends are *defined* to be Bill's cousins.

The end result of this process would be a first-order KB: a collection of sentences in FOL representing what was known about the domain. A major advantage of FOL is that given a yes-no question also expressed in this language, we can give a very precise definition of KB $\models \alpha$ (and thus, under what conditions the question should be answered *yes*, *no*, or *unknown*):

> KB $\models \alpha$ iff every interpretation satisfying all of the sentences in the KB also satisfies α.[6]

There is, moreover, another property of FOL which helps solidify the role of KR. If we assume that the KB is a finite set of sentences and let *KB* stand for their conjunction, it can be shown that

$$\text{KB} \models \alpha \quad \text{iff} \quad \vdash (KB \supset \alpha)$$

In other words, the question as to whether or not the truth of α is implicit in the KB reduces to whether or not a certain sentence is a *theorem* of FOL. Thus, the question-answering operation becomes one of *theorem proving* in FOL.

3.2 The Problem

The good news in reducing the KR service to theorem proving is that we now have a very clear, very specific notion of what the KR system should do; the bad news is that it is also clear that *this service cannot be provided.* The sad fact of the matter is that deciding whether or not a sentence of FOL is a theorem (i.e., the decision problem) is unsolvable. Moreover, even if we restrict the language practically to the point of triviality by eliminating the quantifiers, the decision problem, though now solvable, does not appear to be solvable in anywhere near reasonable time.[7] It is important to realize that this is not a property of particular algorithms that people have looked at but of the *problem* itself: there *cannot* be an algorithm that does the theorem proving correctly in a reasonable amount of time. This bodes poorly, to say the least, for a service that is supposed to be only a part of a larger knowledge-based system.

One aspect of these intractability results that should be mentioned, however, is that they deal with the *worst case* behaviour of algorithms. In practice, a given theorem proving

[6]The assumption here is that the semantics of FOL specify in the usual way what an interpretation is and under what conditions it will satisfy a sentence.

[7]Technically, the problem is now co-NP-complete, meaning that it is strongly believed to be computationally intractable.

algorithm may work quite well. In other words, it might be the case that for a wide range of questions, the program behaves properly, even though it can be shown that there will always be short questions whose answers will not be returned for a very long time, if at all

How serious is the problem, then? To a large extent this depends on the kind of question you would like to ask of a KR subsystem. The worst case prospect might be perfectly tolerable if you are interested in a mathematical application and the kind of question you ask is an open problem in mathematics. Provided progress is being made, you might be quite willing to stop and redirect the theorem prover after a few months if it seems to be thrashing. Never mind worst case behaviour; this might be the *only* case you are interested in.

But imagine, on the other hand, a robot that needs to know about its external world (such as whether or not it is raining outside or where its umbrella is) before it can act. If this robot has to call a KR system utility as a subroutine, the worst case prospect is much more serious. Bogging down on a logically difficult but low-level subgoal and being unable to continue without human intervention is clearly an unreasonable form of behaviour for something aspiring to intelligence.

Not that "on the average" the robot might not do alright. The trouble is that nobody seems to be able to characterize what an "average" case might be like.[8] As responsible computer scientists, we should not be providing a general inferential service if all that we can say about it is that by and large it will probably work satisfactorily. If the KR service is going to be used as a utility and is not available for introspection or control, then it had better be *dependable* both in terms of its correctness and the resources it consumes. Unfortunately, this seems to rule out a service based on theorem proving (in full first-order logic).

3.3 Two Pseudo-solutions

There are at least two fairly obvious ways to minimize the intractability problem. The first is to push the computational barrier as far back as possible. Research in Automatic Theorem Proving has concentrated on techniques for avoiding redundancies and speeding up certain operations in theorem provers. Significant progress has been achieved here, allowing open questions in mathematics to be answered [30,31]. Along similar lines, VLSI

[8]This seems to account more than anything for the fact that there are so few average case results regarding decidability.

and parallel architectural support stands to improve the performance of theorem provers at least as much as it would any search program.

The second way to make theorem provers more usable is to relax our notion of correctness. A very simple way of doing this is to make a theorem proving program always return an answer after a certain amount of time.[9] If it has been unable to prove either that a sentence or its negation is implicit in the KB, it could assume that it was independent of the KB and answer *unknown* (or maybe reassess the importance of the question and try again). This form of error (i.e., one introduced by an incomplete theorem prover), is not nearly as serious as returning a *yes* for a *no*, and is obviously preferrable to an answer that never arrives. This is of course especially true if the program uses its resources wisely, in conjunction with the first suggestion above.

However, from the point of view of KR, both of these are only pseudo-solutions. Clearly, the first one alone does not help us guarantee anything about an inferential service. The second one, on the other hand, might allow us to guarantee an answer within certain time bounds, but would make it very hard for us to specify what that answer would be. If we think of the KR sevice as reasoning according to a certain logic, then the logic being followed is immensely complicated (compared to that of FOL) when resource limitations are present. Indeed, the whole notion of the KR system calculating what is implicit in the KB (which was our original goal) would have to be replaced by some other notion that went beyond the truth theory of the representation language to include the inferential power of a particular theorem proving program. In a nutshell, we can guarantee getting an answer, but not the one we wanted.

One final observation about this intractability is that it is *not* a problem that is due to the formalization of knowledge in FOL. If we assume that the goal of our KR sevice is to calculate what is implicit in the KB, then as long as the truth theory of our representation language is upward-compatible with that of FOL, we will run into the same problem. In particular, using English (or any other natural or artificial language) as our representation language does not avoid the problem as long as we can express in it at least what FOL allows us to express.

[9]The resource limitation here should obviously be a function of how important overall it might be to answer the question.

4 Expressiveness and Tractability

It appears that we have run into a serious difficulty in trying to develop a KR service that calculates what is implicit in a KB and yet does so in a reasonable amount of time. One option we have not yet considered, however, is to *limit* what can be in the KB so that its implications are more manageable computationally. Indeed, as we will demonstrate in this section, much of the research in KR can be construed as trading off expressiveness in a representation language for a more tractable form of inference. Moreover, unlike the restricted dialects of FOL analyzed in the logic and computer science literatures (e.g., in terms of nestings of quantifiers), the languages considered here have at least proven themselves quite useful in practice, however contrived they may appear on the surface.

4.1 Incomplete Knowledge

To see where this tradeoff between expressiveness and tractability originates, we have to look at the use of the expressive power of FOL in KR and how it differs from its use in mathematics.

In the study of mathematical foundations, the main use of FOL is in the formalization of infinite collections of entities. So, for example, we have first-order number and set theories that use quantifiers to range over these classes, and conditionals to state what properties these entities have. This is exactly how Frege intended his formalism to be used.

In KR, on the other hand, the domains being characterized are usually finite. The power of FOL is used not so much to deal with infinities, but to deal with *incomplete knowledge* [19,13]. Consider the kind of facts[10] that might be represented using FOL:

1 ¬Student(john).

> This sentence says that John is not a student without saying what he is.

2. Parent(sue,bill) ∨ Parent(sue,george)

> This sentence says that either Bill or George is a parent of Sue, but does not specify which.

3. $\exists x$ Cousin(bill,x) ∧ Male(x).

> This sentence says that Bill has at least one male cousin but does not say who that cousin is.

[10]The use of FOL to capture *terminology* or laws is somewhat different. See [4] for details.

4. $\forall x$ Friend(george,x) $\supset \exists y$ Child(x,y)

   This sentence says that all of George's friends have children without saying who those friends or their children are or even if there are any.

The main feature of these examples is that FOL is not used to capture complex details about the domain, but to avoid having to represent details that may not be known. *The expressive power of FOL determines not so much what can be said, but what can be left unsaid.*

For a system that has to be able to acquire knowledge in a piecemeal fashion, there may be no alternative to using all of FOL. But if we can restrict the kind of the incompleteness that has to be dealt with, we can also avoid having to use the full expressiveness of FOL. This, in turn, might lead to a more manageable inference procedure.

The last pseudo-solution to the tractability problem, then, is to restrict the logical form of the KB by controlling the incompleteness of the knowledge represented. This is still a pseudo-solution, of course. Indeed, provably, there cannot be a *real* solution to the problem. But this one has the distinct advantage of allowing us to calculate exactly the picture of the world implied by the KB, precisely what a KR service was supposed to do. In what follows, we will show how restricting the logical form of a KB can lead to very specialized, tractable forms of inference.

4.2 Database Form

The most obvious type of restriction to the form of a KB is what might be called *database form.* The idea is to restrict a KB so that it can only contain the kinds of information that can be represented in a standard database. Consider, for example, a very simple database that talks about university courses. It might contain a relation (or record type or whatever) like

COURSE

ID	NAME	DEPT	ENROLLMENT	INSTRUCTOR
csc248	ProgrammingLanguages	ComputerScience	42	S.J.Hurtubise
mat100	HistoryOfMathematics	Mathematics	137	R.Cumberbatch
csc373	ArtificialIntelligence	ComputerScience	853	T.Slothrop
		. . .		

If we had to charaterize in FOL the information that this relation contained, we could use a collection of function-free atomic sentences like[11]

Course(csc248)	Dept(csc248,ComputerScience)	Enrollment(csc248,42)
Course(mat100)	Dept(mat100,Mathematics)	

In other words, the tabular database format characterizes exactly the positive instances of the various predicates. But more to the point, since our list of FOL sentences never ends up with ones like

Dept(mat100,mathematics) $\vee$ Dept(mat100,history),

the range of uncertainty that we are dealing with is quite limited.

There is, however, additional information contained in the database not captured in the simple FOL translation. To see this, consider, for instance, how we might try to determine the answer to the question,

How many courses are offered by the Computer Science Department?

The knowledge expressed by the above collection of FOL sentences is insufficient to answer this question: nothing about our set of atomic sentences implies that Computer Science has at least two courses (since csc373 and csc248 could be names of the same individual), and nothing implies that it has at most two courses (since there could be courses other than those mentioned in the list of sentences). On the other hand, from a database point of view, we could apparently successfully answer our question using our miniature database by phrasing it something like

Count c in COURSE where c.Dept = ComputerScience;

this yields the definitive answer, "2". The crucial difference here, between failing to answer the question at all and answering it definitively, is that we have actually asked ***two different questions***. The formal query addressed to the database must be understood as

[11]This is not the only way to characterize this information. For example, we could treat the field names as function symbols or use *Id* as an additional relation or function symbol. Also, for the sake of simplicity, we are ignoring here integrity constraints (saying, for example, that each course has a unique enrollment), which may contain quantificational and other logical operations, but typically are only used to verify the consistency of the database, not to infer new facts. None of these decisions affect the conclusions we will draw below.

How many tuples in the COURSE relation have ComputerScience in their Dept field?

This is a question *not* about the world being modelled at all, but about the *data* itself. In other words, the database retrieval version of the question asks about the structures in the database itself, and not about what these structures represent.[12]

To be able to reinterpret the database query as the intuitive question originally posed about courses and departments (rather than as one about tuples and fields), we must account for additional information taking us beyond the stored data itself. In particular, we need FOL sentences of the form

$$c_i \neq c_j, \qquad \text{for distinct constants } c_i \text{ and } c_j,$$

stating that each constant represents a unique individual. In addition, for each predicate, we need a sentence similar in form to

$$\forall x[\mathrm{Course}(x) \supset x = \mathrm{csc248} \lor \cdots \lor x \quad \mathrm{mat100}],$$

saying that the only instances of the predicate are the ones named explicitly.[13] If we now consider a KB consisting of all of the sentences in FOL we have listed so far, a KR system could, in fact, conclude that there were exactly two Computer Science courses, just like its Database Management counterpart. We have included in the imagined KB all of the information, both explicit and implicit, contained in the database.

One important property of a KB in this final form is that it is much easier to use than a general first-order KB. In particular, since the first part of the KB (the atomic sentences) does not use negation, disjunction, or existential quantification, we know the exact instances of every predicate of interest in the language. There is no incompleteness in our knowledge at all. Because of this, *inference reduces to calculation*. To find out how many courses there are, all we have to do is count how many appropriate tuples appear in the *COURSE* relation. We do not, for instance, have to reason by cases or by contradiction, as we would have to in the more general case. For example, if we also knew that either *csc148* or *csc149* or both were Computer Science courses but that no Computer Science course other than *csc373* had an odd identification number, we could still determine that

[12]The hallmark, it would appear, of conventional Database Management is that its practitioners take their role to be providing users access to the data, rather than using the data to answer questions about the world. The difference between the two points of view is especially evident when the data is very incomplete [14].

[13]This is one form of what has been called the *closed world assumption* [25].

there were three courses, but not by simply counting. But a KB in database form does not allow us to express this kind of uncertainty and, because of this expressive limitation, the KR service is much more tractable. Specifically, we can represent what is known about the world using just these sets of tuples, exactly like a standard database system. From this perspective, a database is a knowledge base whose limited form permits a very special form of inference.

This limitation on the logical form of a KB has other interesting features. Essentially, what it amounts to is making sure that there is very close structural correspondence between the (explicit) KB and the domain of interest: for each entity in the domain, there is a unique representational object that stands for it; for each relationship that it participates in, there is a tuple in the KB that corresponds to it. In a very real sense, the KB is an *analogue* of the domain of interest, not so different from other analogues such as maps or physical models. The main advantange of having such an analogue is that it can be used directly to answer questions about the domain. That is, the calculations on the model itself can play the role of more general reasoning techniques much the way arithmetic can replace reasoning with Peano's axioms. The disadvantage of an analogue, however, should also be clear: within a certain descriptive language, it does not allow anything to be left unsaid about the domain.[14] In this sense, an analogue representation can be viewed as a special case of a propositional one where the information it contains is relatively complete.

Logic Program Form

The second restriction on the form of a KB we will consider is a generalization of the previous one that is found in programs written in PROLOG, PLANNER, and related languages. A KB in logic program form also has an explicit and an implicit part. The explicit KB in a PROLOG program is a collection of first-order sentences (called Horn sentences) of the form

$$\forall x_1 \cdots x_n [P_1 \wedge \cdots \wedge P_m \supset P_{m+1}] \qquad \text{where} \quad m \geq 0 \text{ and each } P_i \text{ is atomic}$$

In the case where $m = 0$ and the arguments to the predicates are all constants, the logic

[14]The same is true for the standard analogues. One of the things a map does not allow you to say, for example, is that a river passes through one of two widely separated towns, without specifying which. Similarly, a plastic model of a ship cannot tell us that the ship it represents does not have two smokestacks, without also telling us how many it does have. This is not to say that there is no *uncertainty* associated with an analogue, but that this uncertainty is due to the coarseness of the analogue (*e.g.* how carefully the map is drawn) rather than to its content.

program form coincides with the database form. Otherwise, because of the possible nesting of functions, the set of relevant terms (whose technical name is the *Herbrand universe*) is much larger and may be infinite.

As in the database case, if we were only interested in the universe of terms, the explicit KB would be sufficient. However, to understand the KB as being about the world, but in a way that is compatible with the answers provided by a PROLOG processor, we again have to include additional facts in an implicit KB. In this case, the implicit KB is normally infinite since it must contain a set of sentences of the form $(s \neq t)$, for any two distinct terms in the Herbrand universe. As in the database case, it must also contain a version of the closed world assumption which is now a set containing the negation of every ground atomic sentence not implied by the Horn sentences in the explicit KB.

The net result of these restrictions is a KB that once again has complete knowledge of the world (within a given language), but this time, may require inference to answer questions.[15] The reasoning in this case, is the *execution* of the logic program. For example, given an explicit PROLOG KB consisting of

```
parent(bill,mary).
parent(bill,sam).
mother(X,Y) :- parent(X,Y), female(Y)
female(mary).
```

we know exactly who the mother of Bill is, but only after having executed the program.

In one sense, the logic program form does not provide any computational advantage to a reasoning system since determining what is in the implicit KB is, in general, undecidable.[16] On the other hand, the form is much more manageable than in the general case since the necessary inference can be split very nicely into two components: a *retrieval* component that extracts (atomic) facts from a database by pattern-matching and a *search* component that tries to use the non-atomic Horn sentences to complete the inference. In actual systems like PROLOG and PLANNER, moreover, the search component is partially under user control, giving him the ability to incorporate some of the kinds of procedural knowledge (or combinatoric advice) referred to earlier. The only purely automatic inference is the retrieval component.

[15]Notice that it is impossible to state in a KB of this form that $(p \vee q)$ is true without stating which, or that $\exists x P(x)$ is true without saying what that x is. However, see the comments below regarding the use of encodings.

[16]In other words, determining if a ground atomic sentence is implied by a collection of Horn sentences (containing function symbols) is undecidable.

This suggests a different way of looking at the inferential service provided by a KR system (without even taking into account the logical form of the KB). Instead of automatically performing the full deduction necessary to answer questions, a KR system could manage a *limited form of inference* and leave to the rest of the knowledge-based system (or to the user) the responsibility of intelligently completing the inference. As suggested in [10], the idea is to take the "muscle" out of the automatic component and leave the difficult part of reasoning as a problem that the overall system can (meta-)reason about and plan to solve [11].

While this is certainly a promising approach, especially for a KB of a fully general logical form, it does have its problems. First of all, it is far from clear what primitives should be available to a program to extend the reasoning performed by the KR subsystem. It is not as if it were a simple matter to generalize the meager PROLOG control facilities to handle a general theorem prover, for example.[17] The search space in this case seems to be much more complex.

Moreover, it is not clear what the KR service itself should be. If all a KR utility does is perform explicit retrieval over sentences in a KB, it would not be much help. For example, if asked about $(p \vee q)$, it would fail if it only had $(q \vee p)$ in the KB. What we really need is an automatic inferential service that lies somewhere between simple retrieval and full logical inference. But finding such a service that can be motivated *semantically* (the way logical deduction is) and defined independently of how any program actually operates is a non-trivial matter, though one we have taken some steps towards this in [16] (and see

Semantic Network Form

Turning now to semantic networks, a first observation about a KB in this form is that it only contains unary and binary predicates. For example, instead of representing the fact that John's grade in cs100 was 85 by

Grade(john, cs100, 85),

we would postulate the existence of objects called "grade-assignments" and represent the fact about John in terms of a particular grade-assignment *g-a1* as

$$\text{Grade-assignment(g-a1)} \wedge \text{Student(g-a1,john)} \wedge$$
$$\text{Course(g-a1,cs100)} \wedge \text{Mark(g-a1,85)}.$$

[17]Though see [29] for some ideas in this direction

This part of a KB in semantic net form is also in database form: a collection of function-free ground atoms, sentences stating the uniqueness of constants and the closed world assumption.

The main feature of a semantic net (and of the frame form below), however, is not how individuals are handled, but the treatment of the unary predicates (which we will call *types*) and binary ones (which we will call *attributes*). First of all, the types are organized into a taxonomy, which, for our purposes, can be represented by a set of sentences of the form[18]

$$\forall x[B(x) \supset A(x)]$$

The second kind of sentence in the generic KB places a constraint on an attribute as it applies to instances of a type:

$$\forall x[B(x) \supset \exists y(R(x,y) \wedge V(y))] \quad \text{or} \quad \forall x[B(x) \supset R(x,c)].^{19}$$

This completes the semantic net form.

One property of a KB in this form is that it can be represented by a labelled directed graph (and displayed in the usual way). The nodes are either constants or types, and the edges are either labelled with an attribute or with the special label *is*-a.[20] The significance of this graphical representation is that it allows certain kinds of inference to be performed by simple graph-searching techniques. For example, to find out if a particular individual has a certain attribute, it is sufficient to search from the constant representing that individual, up *is*-a links, for a node having an edge labelled with the attribute. By placing the attribute as high as possible in the taxonomy, all individuals below it can *inherit* the property. Computationally, any mechanism that speeds up this type of graph-searching can be used directly to improve the performance of inference in a KB of this form.

In addition, the graph representation suggests different kinds of inference that are based more directly on the structure of the KB than on its logical content. For example, we can ask how two nodes are related and answer by finding a path in the graph between them. Given for instance, Clyde the elephant and Jimmy Carter, we could end up with an

[18]See [2] for a discussion of some of the subtleties involved here.

[19]There are other forms possible for this constraint. For example, we might want to say that *every* R rather than *some* R is a V. See also [12]. For the variant we have here, however, note that the KB is no longer in logic program form.

[20]Note that the interpretation of an edge depends on whether its source and target are constants or types. For example, from a constant c to a type B, *is*-a says $B(c)$, but from a type B to a type A, it is a taxonomic sentence (again, see [2]).

answer saying that Clyde is an elephant and that the favourite food of elephants is peanuts which is also the major product of the farm owned by Jimmy Carter. A typical method of producing this answer would be to perform a "spreading activation" search beginning at the nodes for Clyde and Jimmy. Obviously, this form of question would be very difficult to answer for a KB that was not in semantic net form.[21]

For better or worse, the appeal of the graphical nature of semantic nets has lead to forms of reasoning (such as default reasoning [24]) that do not fall into standard logical categories and are not yet very well understood [9].[22] This is a case of a representational notation taking on a life of its own and motivating a completely different style of use not necessarily grounded in a truth theory. It is unfortunately much easier to develop algorithms that appear to reason over structures of a certain kind than to *justify* its reasoning by explaining what the structures are saying about the world.

This is not to say that defaults are not a crucial part of our knowledge about the world. Indeed, the ability to abandon a troublesome or unsuccessful line of reasoning in favour of a default answer seems intuitively to be a fundamental way of coping with incomplete knowledge in the presence of resource limitations. The problem is to make this intuition precise. Paradoxically, the best formal accounts we have of defaults (such as [26]) would claim that reasoning with them is even *more difficult* than reasoning without them, so research remains to be done.

One final observation concerns the elimination of higher arity predicates in semantic networks. It seems to be fairly commonplace to try to sidestep a certain generality of logical form by introducing special representational objects into the domain. In the example above, a special "grade-assignment" object took the place of a 3-place predicate. Another example is the use of encodings of sentences as a way of providing (what appears to be) a completely extensional version of modal logic [18].[23] Not that exactly the same expressiveness is preserved in these cases; but what *is* preserved is still fairly mysterious and deserves serious investigation, especially given its potential impact on the tractability of inference.

[21]Quillian [23] proposed a "semantic intersection" approach to answering questions in his original work on semantic nets. See also [7] for followup work on the same topic.

[22]A simple example of a default would be to make *elephant* have the colour *grey* but to allow anything below *elephant* (such as *albino-elephant*) to be linked to a different colour value. To determine the colour of an individual would involve searching up for a value and stopping when the first one is found, allowing it to preempt any higher ones. See also [3].

[23]Indeed, some modern semantic network formalisms (such as [27]) actually include all of FOL by encoding sentences as terms.

4.5 Frame Description Form

The final form we will consider, the frame description form, is mainly an elaboration of the semantic network one. The emphasis, in this case, is on the structure of types themselves (usually called *frames*), particularly in terms of their attributes (called *slots*). Typically, the kind of detail involved with the specification of attributes includes

1 *values*, stating exactly what the attribute of an instance should be. Alternatively, the value may be just a *default*, in which case an individual inherits the value provided he does not override it.

2. *restrictions*, stating what constraints must be satisfied by attribute values. These can be *value* restrictions, specified by a type that attribute values should be instances of, or *number* restrictions, specified in terms of a minimum and a maximum number of attribute values.

3. *attached procedures*, providing procedural advice on how the attribute should be used. An *if-needed* procedure says how to calculate attribute values if none have been specified; an *if-added* procedure says what should be done when a new value is discovered.

Like semantic networks, frame languages tend to take liberties with logical form and the developers of these languages have been notoriously lax in characterizing their truth theories [12,3]. What *we* can do, however, is restrict ourselves to a non-controversial subset of a frame language that supports descriptions of the following form:

```
(Student
     with a dept is computer-science and
     with ≥ 3 enrolled-course is a
        (Graduate-Course
             with a dept is a Engineering-Department))
```

This is intended to be a structured type that describes Computer Science students taking at least three graduate courses in departments within Engineering. If this type had a name (say A), we could express the type in FOL by a "meaning postulate" of the form

$$\begin{aligned}\forall x A(x) \equiv [&\text{Student}(x) \wedge \text{dept}(x, \text{computer-science}) \wedge \\ &\exists y_1 y_2 y_3\, (y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3 \wedge \\ &\quad \text{enrolled-course}(x, y_1) \wedge \text{Graduate-Course}(y_1) \wedge \\ &\quad\quad \exists z(\text{dept}(y_1, z) \wedge \text{Engineering-Department}(z)) \wedge \\ &\quad \text{enrolled-course}(x, y_2) \wedge \text{Graduate-Course}(y_2) \wedge \\ &\quad\quad \exists z(\text{dept}(y_2, z) \wedge \text{Engineering-Department}(z)) \wedge \\ &\quad \text{enrolled-course}(x, y_3) \wedge \text{Graduate-Course}(y_3) \wedge \\ &\quad\quad \exists z(\text{dept}(y_3, z) \wedge \text{Engineering-Department}(z)))].\end{aligned}$$

Similarly, it should be clear how to state equally clumsily[24] in FOL that an individual is an instance of this type.

One interesting property of these structured types is that we do not have to state explicitly when one of them is below another in the taxonomy. The descriptions themselves implicitly define a taxonomy of *subsumption*, where type A subsumes type B if, by virtue of the form of A and B, every instance of B must be an instance of A. For example, without any world knowledge, we can determine that the type *Person* subsumes

> (Person **with every** male friend **is a** Doctor)

which in turn subsumes

> (Person **with every** friend **is a**
> (Doctor **with a** specialty **is** surgery)).

Similarly,

> (Person **with** $\geq$ 2 children) subsumes
> (Person **with** $\geq$ 3 male children).

Also, we might say that two types are *disjoint* if no instance of one can be an instance of the other. An example of disjoint types is

> (Person **with** $\geq$ 3 young children) and
> (Person **with** $\leq$ 2 children).

Analytic relationships like subsumption and disjointness are properties of structured types that are not available in a semantic net where all of the types are atomic.

There are very good reasons to be interested in these analytic relationships [4]. In KRYPTON [5,6], a full first-order KB is used to represent facts about the world, but subsumption and disjointness information is also available without having to add to the KB a collection of meaning postulates representing the structure of the types. The reason this is significant is that while subsumption and disjointness can be defined in terms of logical implication,[25] there are good special-purpose algorithms for calculating these relationships in some frame description languages. Again, because the logical form is sufficiently constrained, the required inference can be much more tractable.

[24]What makes these sentences especially awkward in FOL is the number restrictions. For example, the sentence "*There are a hundred billion stars in the Milky Way Galaxy*" would be translated into an FOL sentence with on the order of 10^{22} conjuncts.

[25]Specifically, type A subsumes type B iff the meaning postulates for A and B logically imply $\forall x[B(x) \supset A(x)]$.

An Example of the Tradeoff

As it turns out, frame description languages and the subsumption inference provide a rich domain for studying the tradeoff between expressiveness and tractability. To see this, we will look in some detail at a simple frame description language called $\mathcal{FL}$ whose types and attributes are defined by the following grammar:

$$
\begin{aligned}
\langle type \rangle ::= \ & \langle atom \rangle \\
& \mid (\texttt{AND}\ \langle type_1 \rangle \ldots \langle type_n \rangle) \\
& \mid (\texttt{ALL}\ \langle attribute \rangle\ \langle type \rangle) \\
& \mid (\texttt{SOME}\ \langle attribute \rangle)
\end{aligned}
$$

$$
\begin{aligned}
\langle attribute \rangle ::= \ & \langle atom \rangle \\
& \mid (\texttt{RESTR}\ \langle attribute \rangle\ \langle type \rangle)
\end{aligned}
$$

The $\mathcal{FL}$ language is intended as a simplified (though less readable) version of the frame-based language used in the previous section. So, for example, where we would previously have written a description like

(person **with every** male friend **is a** (doctor **with a** specialty)),

the equivalent $\mathcal{FL}$ type is now written as

(AND person (ALL (RESTR friend male) (AND doctor (SOME specialty)))).

To state exactly what these constructs mean, we now briefly define a straightforward extensional semantics for $\mathcal{FL}$ (which also provides us with a precise definition of subsumption). This will be done as follows: imagine that associated with each description is the set of individuals (individuals for types, pairs of individuals for attributes) that it describes. Call that set the *extension* of the description. Notice that by virtue of the structure of descriptions, their extensions are not independent (for example, the extension of (AND t_1 t_2) should be the intersection of those of t_1 and t_2). In general, the structures of two descriptions can imply that the extension of one is always a superset of the extension of the other. In that case, we will say that the first *subsumes* the second (so, in the case just mentioned, t_1 would be said to subsume (AND t_1 t_2)).

More formally, let $\mathcal{D}$ be any set and $\mathcal{E}$ be any function from types to subsets of $\mathcal{D}$ and attributes to subsets of the Cartesian product, $\mathcal{D} \times \mathcal{D}$. So

$$\mathcal{E}[t] \subseteq \mathcal{D} \quad \text{for any type } t \text{, and}$$
$$\mathcal{E}[a] \subseteq \mathcal{D} \times \mathcal{D} \quad \text{for any attribute } a.$$

We will say that $\mathcal{E}$ is an *extension function* over $\mathcal{D}$ if and only if

1. $\mathcal{E}[(\texttt{AND}\ t_1 \quad t_n)] \qquad \bigcap_i \mathcal{E}[t_i]$

2. $\mathcal{E}[(\texttt{ALL}\ a\ t)] = \left\{x \in \mathcal{D} \mid \text{if} \quad \langle x,y\rangle \in \mathcal{E}[a] \quad \text{then} \quad y \in \mathcal{E}[t]\right\}$

3. $\mathcal{E}[(\texttt{SOME}\ a)] = \left\{x \in \mathcal{D} \mid \exists y \left[\langle x,y\rangle \in \mathcal{E}[a]\right]\right\}$

4. $\mathcal{E}[(\texttt{RESTR}\ a\ t)] = \left\{\langle x,y\rangle \in \mathcal{D} \times \mathcal{D} \quad \langle x,y\rangle \in \mathcal{E}[a] \ \text{and} \ y \in \mathcal{E}[t]\right\}$

Finally, for any two types t_1 and t_2, we can say that t_1 *is subsumed by* t_2 if and only if for any set $\mathcal{D}$ and any extension function $\mathcal{E}$ over $\mathcal{D}$, $\mathcal{E}[t_1] \subseteq \mathcal{E}[t_2]$. That is, one type is subsumed by a second type when all instances of the first—in all extensions—are also instances of the second. From a semantic point of view, subsumption dictates a kind of necessary set inclusion.

Given a precise definition of subsumption, we can now consider algorithms for calculating subsumption between descriptions. Intuitively, this seems to present no real problems. To determine if s subsumes t, what we have to do is make sure that each component of s is "implied" by some component (or components) of t. Moreover, the type of "implication" we need should be fairly simple since $\mathcal{FL}$ has neither a negation nor a disjunction operator.

Unfortunately, such intuitions can be nastily out of line. In particular, let us consider a slight variant of $\mathcal{FL}$—call it $\mathcal{FL}^-$. $\mathcal{FL}^-$ includes all of $\mathcal{FL}$ except for the RESTR operator. On the surface, the difference between $\mathcal{FL}^-$ and $\mathcal{FL}$ seems expressively minor. But it turns out that it is computationally very significant. In particular, there is an $O(n^2)$ algorithm for determining subsumption in $\mathcal{FL}^-$, but the same problem for $\mathcal{FL}$ is intractable. In the rest of this section, we sketch the form of our algorithm for $\mathcal{FL}^-$ and the proof that subsumption for $\mathcal{FL}$ is as hard as testing for propositional tautologies, and therefore most likely unsolvable in polynomial time[26]

We here simply present, without comment, an algorithm for computing subsumption for $\mathcal{FL}^-$:

Subsumption Algorithm for $\mathcal{FL}^-$ SUBS?[s,t]

1. Flatten both s and t by removing all nested AND operators. So, for example,

   (AND x (AND y z) w) becomes (AND x y z w)

2. Collect all arguments to an ALL for a given attribute. For example,

[26]Details of all proofs can be found in [17

(AND (ALL a (AND u v w)) x (ALL a (AND y z))) becomes
(AND x (ALL a (AND u v w y z))).

3 Assuming s is now (AND s_1 s_n) and t is (AND t_1 t_m), then return **true** iff for each s_i,

(a) if s_i is an atom or a SOME, then one of the t_j is s_i

(b) if s_i is (ALL a x), then one of the t_j is (ALL a y), where SUBS?[x,y].

This algorithm can be shown to compute subsumption correctly. For the purposes of this paper, the main property of SUBS? that we are interested in is that it can be shown to calculate subsumption for $\mathcal{FL}^-$ in $O(n^2)$ time.

We now turn our attention to the subsumption problem for full $\mathcal{FL}$. The proof that subsumption of descriptions in $\mathcal{FL}$ is intractable is based on a correspondence between this problem and the problem of deciding whether a sentence of propositional logic is implied by another. Specifically, we define a mapping π from propositional sentences in conjunctive normal form to descriptions in $\mathcal{FL}$ that has the property that for any two sentences α and β, α logically implies β iff $\pi[\alpha]$ is subsumed by $\pi[\beta]$.

Suppose $p_1, p_2, \ldots, p_m$ are propositional letters distinct from A, B R, and S.

```
π[p₁ ∨ p₂ ∨ ... ∨ pₙ ∨ ¬pₙ₊₁ ∨ ¬pₙ₊₂ ∨ ... ∨ ¬pₘ]
    (AND (ALL (RESTR R p₁) A)
          ...
         (ALL (RESTR R pₙ) A)
         (SOME (RESTR R pₙ₊₁))
          ...
         (SOME (RESTR R pₘ)))
```

Assume that $\alpha_1, \alpha_2, \ldots, \alpha_k$ are disjunctions of literals not using A, B, R, and S.

```
π[α₁ ∧ α₂ ∧ ... ∧ αₖ] =
    (AND (ALL (RESTR S (SOME (RESTR R A))) B)
         (ALL (RESTR S π[α₁]) B)

         (ALL (RESTR S π[αₖ]) B))
```

What this mapping provides is a way of answering questions of implication by first mapping the two sentences into descriptions in $\mathcal{FL}$ and then seeing if one is subsumed by the other. Moreover, because π can be calculated efficiently, any *good* algorithm for subsumption becomes a good one for implication.

The key observation here, however, is that there can be no good algorithm for implication. To see this, note that a sentence implies $(p \wedge \neg p)$ just in case it is not satisfiable. But determining the satisfiablity of a sentence in this form is NP-complete [8]. Therefore, a special case of the implication problem (where the second argument is $(p \wedge \neg p)$) is the complement of an NP-complete one. The correspondence between implication and subsumption, then, leads to the observation that subsumption for $\mathcal{FL}$ is co-NP hard. In other words, since a good algorithm for subsumption would lead to a good one for implication, subsumption over descriptions in $\mathcal{FL}$ is intractable.[27]

5 Conclusions and Morals

In this final section, we step back from the details of the specific representational formalisms we have examined and attempt to draw a few conclusions.

An important observation about these formalisms is that we cannot really say that one is *better* than any other; they simply take different positions on the tradeoff between expressiveness and tractability. For example, full FOL is both more expressive and less appealing computationally than a language in semantic net form. Nor is it reasonable to say that expressiveness is the primary issue and that the other is "merely" one of efficiency. In fact, we are not really talking about efficiency here at all; that, presumably, is an issue of algorithm and data structure, concerns of the Symbol Level [21]. The tractability concern we have here is much deeper and involves whether or not it makes sense to even think of the language as computationally based.

From the point of view of those doing research in KR, this has a very important consequence: we should continue to design and examine representation languages, *even when these languages can be viewed as special cases of FOL*. What really counts is that these special cases be interesting both from the point of view of what they can represent, and from the point of view of the reasoning strategies they permit. All of the formalisms we have examined above satisfy these two requirements. To dismiss a language as *just* a

[27] As mentioned in section 3.2, the co-NP-complete problems are strongly believed to be unsolvable in polynomial time.

subset of FOL is probably as misleading as dismissing the notion of a context-free grammar as just a special case of a context-sensitive one.

What truth in advertising does require, however, is that these special cases of FOL be identified as such. Apart from allowing a systematic comparison of representation languages (as positions on the tradeoff), this might also encourage us to consider systems that use more than one sublanguage and reasoning mechanism (as suggested for equality in [20]). The KRYPTON language [5,6], for example, includes all of FOL *and* a frame description language. To do the necessary reasoning, the system contains both a theorem prover and a description subsumption mechanism, even though the former could do the job of the latter (but much less efficiently). The trick with these *hybrid systems* is to factor the reasoning task so that the specialists are able to cooperate and apply their optimized algorithms without interfering with each other.

These considerations for designers of representation languages apply in a similar way to those interested in populating a KB with a theory of some sort. A good first step might be to write down a set of first-order sentences characterizing the domain, but it is somewhat naive to stop there and claim that the account could be made computational after the fact by the inclusion of a theorem prover and a few well chosen heuristics. What is really needed is the (much more difficult) analysis of the logical form of the theory, keeping the tradeoff clearly in mind. An excellent example of this is the representation of *time* described in [1]. Allen is very careful to point out what kind of information about time cannot be represented in his system, as well as the computational advantage he gains from this limitation.

For the future, we still have a lot to learn about the tradeoff. It would be very helpful to accumulate a wide variety of data points involving tractable and intractable languages. Especially significant are crossover points where small changes in a language change its computational character completely (an instance of which was illustrated in section 4.6). Moreover, we need to know more about what *people* find easy or hard to handle. There is no doubt that people can reason when necessary with radically incomplete knowledge (such as that expressible in full FOL) but apparently only by going into a special problem-solving or logic puzzle mode. In normal common sense situations, when reading a geography book, for instance, the ability to handle disjunctions (say) seems to be quite limited. The question is what forms of incomplete knowledge *can* be handled readily, given that the geography book is not likely to contain any procedural advice on how to reason.

In summary, we feel that there are many interesting issues to pursue involving the tradeoff between expressiveness and tractability. Although there has always been a temptation in KR to set the sights either too low (and provide only a data structuring facility with little or no inference) or too high (and provide a full theorem proving facility), this paper argues for the rich world of representation that lies between these two extremes.

Acknowledgements

Many of the ideas presented here originally arose in the context of the KRYPTON project. We are deeply indebted to the Fairchild AI Lab (now the AI Lab of Schlumberger Computer Aided Systems Research) for making this research possible and to Richard Fikes, Peter Patel-Schneider, and Victoria Gilbert. We would especially like to thank Peter, and also Jim des Rivières for providing very helpful comments on a earlier draft of this paper, and S. J. Hurtubise for not.

Bibliography

[1] Allen, J., "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, Vol. 26, No. 11, November, 1983, 832–843.

[2] Brachman, R. J., "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks," *IEEE Computer*, Vol. 16, No. 10, October, 1983, 30–36.

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

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

[5] Brachman, R. J., Fikes, R. E., and Levesque, H. J., "Krypton: A Functional Approach to Knowledge Representation," *IEEE Computer*, Vol. 16, No. 10, October, 1983, 67–73.

[6] Brachman, R. J., Gilbert, V. P., and Levesque, H. J., "An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON," *Proc. IJCAI-85*, Los Angeles, CA, August, 1985.

[7] Collins, A. M., and Loftus, E. F., "A Spreading-activation Theory of Semantic Processing," *Psychological Review*, Vol. 82, No. 6, 1975, 407–428.

[8] Cook, S. A., "The Complexity of Theorem-Proving Procedures," *Proc. 3rd Ann. ACM Symposium on Theory of Computing*. New York: Association for Computing Machinery, 1971, pp. 151–158.

[9] Etherington, D., and Reiter, R., "On Inheritance Hierarchies with Exceptions," *Proc. AAAI-83*, Washington, DC, August, 1983, 104–108.

[10] Frisch, A., and Allen, J., *Knowledge Representation and Retrieval for Natural Language Processing*, TR 104, Computer Science Dept., Univ. of Rochester, 1982.

[11] Genesereth, M. R., "An Overview of Meta-Level Architecture," *Proc AAAI-83*, Washington, DC, August, 1983, 119–123.

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

[13] Levesque, H. J., *A Formal Treatment of Incomplete Knowledge Bases*, Technical Report No. 3, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA, 1982.

[14] Levesque, H. J., "The Logic of Incomplete Knowledge Bases," in *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. Schmidt (eds.). New York: Springer-Verlag, 1984, 165–186.

[15] Levesque, H. J., "Foundations of a Functional Approach to Knowledge Representation," *Artificial Intelligence*, Vol. 23, No. 2, July, 1984, 155–212.

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

[17] Levesque, H. J., and Brachman, R. J., "Some Results on the Complexity of Subsumption in Frame-Based Description Languages," in preparation.

[18] Moore, R. C., *Reasoning about Knowledge and Action*, Technical Note 191, SRI International, Menlo Park, CA, 1980.

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

[20] Nelson, G., and Oppen, D. C., "Simplification by Cooperating Decision Procedures," *ACM Transactions on Programming Languages and Systems*, Vol. 1, No. 2, 1979, 245–257.

[21] Newell, A., "The Knowledge Level," *The AI Magazine*, Vol. 2, No. 2, Summer, 1981, 1–20.

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

[23] Quillian, M. R., "Semantic Memory," in *Semantic Information Processing*, M. Minsky (ed.). Cambridge, MA: MIT Press, 1968, 227–270.

[24] Reiter, R., "On Reasoning by Default," *Proc. TINLAP-2*, University of Illinois at Urbana-Champaign, 1978, 210–218.

Reiter, R., "On Closed World Data Bases," in *Logic and Data Bases*, H. Gallaire and J. Minker (eds.). New York: Plenum Press, 1978, 55–76.

Reiter, R., "A Logic for Default Reasoning," *Artificial Intelligence*, Vol. 13, Nos. 1,2 April, 1980, 81–132.

[27] Shapiro, S. C., "The SNePS Semantic Network Processing System," in *Associative Networks: Representation and Use of Knowledge by Computers*, N. V. Findler (ed.). New York: Academic Press, 1979, 179–203.

[28] Smith, B. C., *Reflection and Semantics in a Procedural Language*, Ph.D. Thesis and Tech. Report MIT/LCS/TR-272, MIT, Cambridge, MA, 1982.

[29] Stickel, M. E., "A Prolog Technology Theorem Prover," *Proc. of the 1984 Symposium on Logical Programming*, Atlantic City, 1984, 211–217.

[30] Winker, S., "Generation and Verification of Finite Models and Counterexamples using an Automated Theorem Prover Answering Two Open Questions," *Journal of the ACM*, Vol. 29, No. 2, April, 1982, 273–284.

[31] Wos, L., Winker, S., Smith, B., Veroff, R., and Henschen, L., "A New Use of an Automated Reasoning Assistant: Open Questions in Equivalential Calculus and the Study of Infinite Domains," *Artificial Intelligence*, Vol. 22, No. 3, April, 1984, 303–356.