A LOGICAL MODEL OP KNOWLEDGE

Martin Nilsson
UPMAIL
Uppsala Programming Methodology and Artificial Intelligence Laboratory
Department of Computing Science, Uppsala University
P.O. Box 2050, S - 750 02 Uppsala, Sweden

Abstract

There are several approaches to formalizing knowledge. The concept of knowledge is often introduced by a special operator whose properties are defined by a number of axioms. A situation is then modelled by using this operator in problem-specific axioms. This article presents a different approach - a first-order logical model which does not use any new operators. The properties of knowledge should follow from the problem-specific axioms only. The advantages are that the model is simpler and requires fewer axioms. The formalization of a problem can almost directly be used as a computer program which produces the solutions.

Keywords: Epistemology, Knowledge, FirstrOrder Logic, False Knowledge, Problem Solving, Logic Programming.

1. Introduction

The classical work in the field of formalizing knowledge is by Hintikka [1]. McCarthy and Hayes [6] point to the importance of epistemology for Artificial Intelligence. Several different epistemological theories with connections to computer science have followed, e.g. by McCarthy [4], Moore [7], Konolige [2], and Levesque (3).

All of the authors [1,2,3,4,7] model knowledge by introducing a truth-valued operator *K.* This operator is described by a number of axioms. The knowledge that a formula holds is formalized by saying that *K* holds for an argument that is a term which represents that formula. A drawback of this method is that a large number of axioms are required to describe *K,* and to define the mapping between terms and formulas. Another hard problem is the problem of substituting equals for equals in the arguments of a *referentially* opaque operator like *K.* Following an example by Aristotle: The man approaching you is Coriscus. You know that Coriscus is musical, but that does not imply that you know that the approaching man is musical, even though the men are identical. *You* have to *know* that they are identical. There are plenty of other problems with models of knowledge, and they are carefully studied in the impressively thorough work [7].

We argue that too many axioms will make the computational treatment of knowledge more complicated than necessary. They force the user to have a general theorem prover, they increase the indeterminacy of the computation, etc. The main difference between the above models and our model is that it does not have the intermediate step of a knowledge operator. The formalization of a problem is self-contained (except for the logical axioms of first-order logic). It has a form which greatly simplifies use of the formalization as a computer program.

The research reported herein was supported by the Swedish board for technical development (STU).

2. Philosophical Mews on Knowledge

We consider the sources of knowledge to be the following: perceived information, information inferred from other knowledge, and previous knowledge. The procedures for inferring knowledge are unspecified. As an idealization we shall assume that they are logical deductions. Obviously, a person cannot know all the logical implications of everything he knows, but for solving most problems he will not really need to.

It should be noted that we use the AI view of knowledge, which in terms of classical philosophy is closer to belief. This is because false knowledge might exist in the system: For instance, an agent might perceive false information. Applications of our model to false knowledge is further treated in Nilsson (8|.

3. Notation

Variables are in lower case, and constants have an initial upper case letter. Universal quantifiers are often omitted at the top level of formulas. Our standard formalism is first-order logic, but we will often use set *notation* to make our formulas easier to read. However, this is only a kind of syntactic sugar, and it should always be clear how to translate back to logic. For a predicate w, we imagine a set *W* such that *x* is a member of *W* iff w(x). For instance, the following formulas say - in logic and in set notation - that there is more than one *x* such that w(x), that there is exactly one *x* such that w(x), and that there is no *x* such that w(x), respectively:

$$
\begin{array}{ll}
\exists x \exists y (w(x) \wedge w(y) \wedge x \neq y) & W = \{\_, \_, \ldots\} \\
\exists x \forall y (w(x) \wedge (w(y) \rightarrow x = y)) & W = \{\_\} \\
\neg \exists x \, w(x) & W = \{\}
\end{array}
$$

A predicate $w(x_1, \ldots, x_n, x_{n+1})$ depending on $n+1$ arguments corresponds analogously to a set function $W(x_1, \ldots, x_n)$ of n arguments.

The notation $f : W$ is used for the image of a set W under the mapping f, or $f : W = \{f(x) \mid w(x)\}$, the set of all $f(x)$ such that $w(x)$. Logically, this set corresponds to a predicate $w_f(x) \leftrightarrow \exists y(w(y) \wedge x = f(y))$. Sometimes it is convenient to look at predicates as functions ranging over the set of truth values, $\{True, False\}$. For instance, $w : W = \{True\}$. The set $p : W$ then corresponds to a predicate $w_p(x) \leftrightarrow \exists y(w(y) \wedge x = f_p(y))$, where f_p is a function: $f_p(y) = True \leftrightarrow p(y)$, $f_p(y) = False \leftrightarrow \neg p(y)$, where *True* and *False* are constants representing truth values.

4. A Model of Questions

The theory of knowledge relates closely to the theory of questions. Knowledge can be seen as having the answer to a question, or the solution to a problem. We will first model questions using a possible worlds approach. Then we will extend that model to cover knowledge. We use the phrase "possible worlds" because it is intuitively descriptive. We do not intend to automatically inherit a modal type of logic, or properties of such a logic, which are often associated with possible worlds.

For every problem we have a domain W_0, which is the set of possible worlds or descriptions of states. One of these is the actual world, or a description of the current state. The set of all possible solutions W_1 is the set of all members x of W_0 such that some problem-specific constraint p(x) holds. Sometimes we might need to generalize this idea a bit: The set of solutions is instead the set of function values / : W_l where W_1 is the set of all states in W_0 satisfying p(x). The function / extracts the interesting information from the state description.

Traditional philosophy classifies questions as "whether" questions and "what" questions. The former kind asks for a truth value, "Does p hold?" The latter kind asks for a value, "For which x does p(x) hold?" If predicates are regarded as truth valued functions, we can cover both of these cases by a more general question: "What is the value of *f(x)* when p(x)?" In the first case, the function / is equivalent to p, and in the second, it is the identity function. Let *x* be the actual world, i.e. a description of the current state. Then the actual answer to the question above will be /(x). It is important to note here that the actual answer is supposed to be unique. The question "What is a square root of 1?" should therefore be reformulated as "What are the square roots of 1?" The domain could then be the set of sets of numbers, and the solution is the set {-1,1}. We cannot choose the set of numbers as the domain, since that does not give us a sufficient state representation.

5. A Model of Knowledge

We shall now model knowledge in terms of the number of possible solutions to a question.

If we describe knowledge by "having the answer to a question," we can formalize it by saying that the set of possible solutions contains exactly one element, namely the actual world. If the set of possible solutions contains more than one element, we can't know which one corresponds to the actual world, and therefore we don't know the solution. Thus our basic models of knowledge are

Do not know f	$f : W = \{\_, \_, \ldots\}$
Do know f	$f : W = \{\_\}$

where *W* is the set of possible worlds.

Conditions such as "*A* knows that *B* knows /," where *A* and *B* are two different agents, can now easily be formulated: Let *W{A)* be A's set of possible worlds, and let *W(B,x)* be B's set of possible worlds seen from A's viewpoint. This set depends on the actual world z. Since *A* does not necessarily know what the actual world is, he has to try out all his possible worlds for x.

$$p : W(A) = \{True\}.$$
$$p(x) \leftrightarrow f : W(B, x) = \{\_\}.$$

The first formula here says that p holds for all elements of *W(A),* i.e. *A* knows that p. The second defines p(x) to hold iff *A* computes / : *W(B,* x) to contain exactly one element, i.e. if *A* can deduce that *B* must know /, assuming that the actual world is x. For comparison, using logic notation, equivalent formulas are:

$$\exists x \forall y (w(A, x) \wedge (w(A, y) \rightarrow (p(x) \leftrightarrow p(y))) \wedge p(x))$$
$$p(x) \leftrightarrow \exists y \forall z (w(B, x, y) \wedge (w(B, x, z) \rightarrow f(y) = f(z)))$$

The different sets of possible worlds usually depend on two arguments - the agent and the actual world (or what is assumed to be the actual world for the moment) - and the corresponding predicates depend on the same two and also the possible world to be tested. The predicate w here is in fact very similar to the accessibility relation *K* of [7]. Here, however, no special axiom is supposed to bold for w.

"I know" is represented with a singleton, and " I don't know" with a set of more than one element. It is natural to ask if the empty set equation / : *W* = {} has any meaning. / : *W* is empty iff *W* is empty. Erroneous information or an incorrect assumption has made the agent rule out the actual world from his set of possible worlds. ([7]) prohibits this exclusion by a reflexivity axiom which in our formalism would be VaVx w(a, 2, x).) This corresponds to false questions in the theory of questions. A classical example is "Have you stopped beating your wife yet?" This question contains false premises (hopefully), and has no answer. Here, neither of "I know that you have stopped" nor "I don't know if you have stopped" is adequate. A third case, which can be interpreted as a detection of a false premise, must be included. This view seems to be adequate particularly for problems about false knowledge |8|.

6. Examples of Properties of Knowledge

The deductions in this section are somewhat informal for the sake of space. An intuitive property of knowledge is that "I know f" implies that "I know that I know /." In our model the first statement is

$$f : W = \{\_\}$$

If we define p(x)<->/ : *W* = {.} (where *W* is independent of x) we can express the statement by Vzp(z). But since *W* is non-empty, we must have

$$p : W = \{True\}$$

which simply says that "I know that I know f"

Another example is Aristotle's Coriscus-problem. One possible world representation here is a tuple *of* boolean values, *<musicality-of-Coricu, muricality-of~man>.* If *we* define

$p_1(< x, y >) \leftrightarrow x = True$	**(Coriscus is musical)**
$p_2(< x, y >) \leftrightarrow y = True$	**(The man is musical),**

the fact that it is known that Coriscus is musical becomes

(A) $\quad p_1 : W = \{True\}$

The knowledge that the man is musical becomes

(B) $\quad p_2 : W = \{True\}$

p_1, restricts *W* to $\{< True, True >, < True, False >\}$. Now, (B) holds only if *W* is further restricted by a constraint that implies that the two components are equal, or, in other words, that it is *known* that they are equal.

Suppose that it is known that p holds, and that it is known that p —► q holds. We should then be able to deduce that it is known that q holds. But for all x, p(x) and p(x) —► q(x) imply that q(x). In particular, the antecedents hold for all z in *W,* and so q(x) holds for all x in *W.* Since *W* is non-empty, q : *W* = {-}.

7. Some Results from Applying the Model to Problem Solving

McCarthy [5] has suggested a certain kind of problem as a test of the adequacy of models of knowledge. These problems deal with knowledge about knowledge and are often hard to solve because of their complicated structure. We have applied oar model to several such non-toy problems, among others the well-known "Mr. S. and Mr. P." problem (Freudenthal, 1900). It has turned out that with our model we can solve these problems by computer in a simple way. The modelling and solving methods are described in detail in [8|. Many problems dealing with false knowledge can be handled within first-order logic without risking non-monotonicity. For a large class of problems 1) the formalization is very similar to the original problem, and 2) the formalization can almost directly be used as computer program which generates the solutions. Due to the stereotypical form of the formalizations, they are extremely easy to translate to a logic programming language, for instance Prolog.

The backtracking depth-first strategy of Prolog leads to an "enumerate and test" solution process. This strategy is sometimes very inefficient (especially when the domain is infinite), although the program in many cases can be rewritten or transformed to a more efficient form. Future research will have to find out if and how that can be done automatically.

8. Acknowledgments

Gunnar Blomberg's help to typeset this article is much appreciated. Also, I want to thank my colleagues at UPMAIL for useful comments to earlier drafts of this paper.

9. Appendix

To exemplify, we shall give a formalization of the "Mr. S. and Mr. P." problem. The formalization is followed by a program that solves the problem.

The problem goes as follows: A person thinks of two numbers between 2 and 99, inclusive. (This defines the domain.) He then tells Mr. S. the sum (constraint p_1), and Mr. P. the product (p_2). A dialogue starts between Mr. S. and Mr. P. "I don't know the numbers," says Mr. P. (p_3). "I don't know them either, but I knew that you wouldn't know," says Mr. S. (p_4). "Then I know them!" answers Mr. P. (p_5). "Then I know them too," says Mr. S. (p_6).

The w:s in the formalization represent the successive sets of solutions, which are restricted by the constraints p_i. The set of solutions is the final set of possibilities from the problem solver's point of view, $W(O_1, s)$, or the set of z such that $w(O_1, s, z)$ holds.

$$w(0, s, (m, n)) \leftrightarrow m \in \{2, \ldots, 99\} \wedge n \in \{2, \ldots, m\}.$$
$$w(S_1, s, z) \leftrightarrow w(0, s, z) \wedge p_1(s, z).$$
$$w(P_1, s, z) \leftrightarrow w(0, s, z) \wedge p_2(s, z).$$
$$w(S_2, s, z) \leftrightarrow w(S_1, s, z) \wedge p_3(z).$$
$$w(P_2, s, z) \leftrightarrow w(P_1, s, z) \wedge p_4(z).$$
$$w(S_3, s, z) \leftrightarrow w(S_2, s, z) \wedge p_5(z).$$
$$w(O_1, s, z) \leftrightarrow w(0, s, z) \wedge p_3(z) \wedge p_4(z) \wedge p_5(z) \wedge p_6(z).$$
$$p_1((m, n), (i, j)) \leftrightarrow m + n = i + j.$$
$$p_2((m, n), (i, j)) \leftrightarrow mn = ij.$$
$$p_3(z) \leftrightarrow \{y \mid w(P_1, z, y)\} = \{\_, \_, \ldots\}.$$
$$p_4(z) \leftrightarrow \{y \mid w(S_2, z, y)\} = \{\_, \_, \ldots\} \wedge \{p_3(z) \mid w(S_1, z, z)\} = \{True\}.$$
$$p_5(z) \leftrightarrow \{y \mid w(P_2, z, y)\} = \{\_\}.$$
$$p_6(z) \leftrightarrow \{y \mid w(S_3, z, y)\} = \{\_\}.$$

The Prolog program for the problem is written in Dec-10 Prolog dialect [9]. The answers are generated by asking the system "? − w(o1, _, X)."

```
w(0, S, (M, N)) :- inclosed(M, [2, 99]), inclosed(N, [2, M]).
inclosed(I, [J, L]) :-
        J =< L, (I = J; K is J + 1, inclosed(I, [K, L])).
value(X, true) :- call(X).
value(X, false) :- \+call(X).
w(s1, S, X) :- w(0, S, X), p1(S, X).
w(p1, S, X) :- w(0, S, X), p2(S, X).
w(s2, S, X) :- w(s1, S, X), p3(X).
w(p2, S, X) :- w(p1, S, X), p4(X).
w(s3, S, X) :- w(s2, S, X), p5(X).
w(o1, S, X) :- w(0, S, X), p3(X), p4(X), p5(X), p6(X).
p1((M, N), (I, J)) :- M + N =:= I + J.
p2((M, N), (I, J)) :- M * N =:= I * J.
p3(X) :- setof(Y, w(p1, X, Y), [_, _, ..._]).
p4(X) :- setof(Y, w(s2, X, Y), [_, _, ..._]),
        setof(Z, W ^ (w(s1, X, W), value(p3(W), Z)), [true]).
p5(X) :- setof(Y, w(p2, X, Y), [_]).
p6(X) :- setof(Y, w(s3, X, Y), [_]).
```

The efficiency of this Prolog program can be raised if we replace the calk to *set of* by some procedure that does not generate more instances of the first argument than necessary. Also, the definition of *w(sl,S,X)* can be made more efficient by integrating w(0, 5, *X)* and pl(S,X), and similarly for *w(pl,* S,*X).* The CPU-time required to solve the problem on a Dec-2060 is then on the order of 20 minutes.

10. References

[l] Hintikka, J.: *Knowledge and Belief: An Introduction to the Logic of the Two Notions.* Cornell University Press, Ithaca, New York, 1062.

[2] Konolige, K.: *A First-Order Formalization of Knowledge and Action for a Multi-Agent Planning System.* Technical Note 232, SRI International, Menlo Park, California, 1080.

(3) Levesque, H. J.: *The Interaction with Incomplete Knowledge Bases: A Formal Treatment.* In Proc. IJCAI-81. Vancouver, B.C., Canada, August, 1081, Vol. 1, pp. 240 245.

(4) McCarthy, J.: *First Order Theories of Individual Concepts and Propositions.* In J.E. Hayes & D. Michie & L.I. Mikulich: Machine Intelligence 0. Ellis Horwood Ltd, 1070, pp. 120-147.

[5| *"Challenge'* attributed to John McCarthy. AISB Quarterly, Issue 35, October, 1070, p. 38.

|6] McCarthy, J. & P.J. Hayes: *Some Philosophical Problems from the Standpoint of Artificial Intelligence.* In B. Meltzer & D. Michie: Machine Intelligence 4. Edinburgh University Press, Edinburgh, 1060, pp. 463-502.

[7] Moore, R.: *Reasoning about Knowledge and Action.* Technical Note 101, SRI International, Menlo Park, California, 1080.

|8| Nilsson, M.: *A Logical Model of Knowledge that Handles False Knowledge.* Technical Report (in preparation). UPMAIL, Computing Science Department, Uppsala University, Sweden, 1083.

[9] Pereira, L. M. & F. Pereira & D. Warren: *User's Guide to DEC system-10 Prolog.* Laboratorio National de Engenharia Civil. Lisbon. Portugal. 1070.