An Epistemic Logic with Quantification over Names

Andrew R. Haas
Department of Computer Science
The University at Albany
1400 Washington Avenue
Albany, New York 12222
(518) 442-4280

August 18, 1993

1 The Problem of Quantifying In

1.1 Konolige's Solution

A sentential theory of attitudes holds that propositions (the things that agents believe and know) are sentences of a representation language. The idea has an obvious appeal to AI workers, who rely heavily on representation languages. Moore and Hendrix (1982) gave the classic statement of the case for a sentential theory of attitudes. Haas (1986), Perlis (1988), and Morgenstern (1987) are among the authors who have developed sentential theories and applied them to problems in AI.

Konolige (1986) proposed a resolution theorem proving algorithm for his version of the sentential theory, and he proved that this algorithm was sound and complete. This is the only known technique for reasoning efficiently in a sentential theory of attitudes. Not surprisingly, he had to limit the expressive power of his logic in order to achieve efficiency. I will criticize his treatment of one problem: quantification into the scope of attitudes. I will argue that in this area Konolige's logic is clearly too weak. In the next section I will sketch a new logic that overcomes the limitations of Konolige's system.

Quantification into the scope of attitudes is a difficult problem for any theory of attitudes. The problem arises when a quantifier stands outside the scope of an attitude operator, and binds a variable that appears inside the scope of that operator. Suppose John knows who the president of IBM is. We might try to represent this information as follows.

[1] $(\exists p.know(john, president(ibm, p)))$

That is, there exists a person p such that John knows that p is the president of IBM. How can we assign a formal semantics to this sentence? It tells us that John knows a proposition about a person p. If that proposition is a sentence of a representation language, it must contain some name for the person p. A representation language might easily include several names for the same person. Which of these names appears in the sentence John knows?

Kaplan (1975) was perhaps the first to consider this question. He tried to define a class of privileged names, and he gave a semantics for sentence (1) by requiring that a privileged name for p must appear in the sentence John knows. Definining this privileged class of names is difficult, but AI workers using sentential theories have largely ignored these difficulties. Morgenstern (1987) tells us that "@ applied to an object yields the name of that object", as if it were obvious that every object has a unique name. Konolige (1986) assumes that every object has a unique privileged name, but he does not tell us why one name is favored over another.

There is no immediate prospect of a general, domain-independent definition of the class of privileged names. For AI workers, the obvious alternative is to study the problem in particular, limited domains. For this purpose it would be useful to have an epistemic logic that allows the user to define the classes of privileged names that arise in a particular domain. The present paper will describe such a logic. This logic does not solve the problem of privileged names, but it provides a tool for working on the problem. For purposes of comparison, let us first consider Konolige's treatment of the problem in more detail.

Konolige introduced a *naming map*: a function that takes an agent a and an element x of the domain of discourse, and returns a constant of the representation language that denotes x. This is the constant that agent a uses as a name for the entity x in his beliefs, and it is called an *id constant* for x. Given the naming map, it is easy to define the semantics of sentence (1). For any value of p, the wff $know(john, president(ibm, p))$ is true iff John knows the sentence $president(ibm, c)$, where c is John's id constant for the value of p.

This approach has several drawbacks. First, the id constants are always constants – never complex terms. Second, there is only one naming map. Most important, the logic does not allow us to describe the naming map. We cannot assert that John's id constant for Bill is the constant c. The user of the logic cannot define a new naming map that arises in a particular domain. Let us consider

some examples, which show how these limitations will hamper us in describing everyday knowledge.

A phone number is a list of seven integers between 0 and 9 (ignoring the area code). Suppose John knows the sentence

[2] $phone(mary, [4,4,2,4,2,8,0])$

(We are using Prolog's notation for lists.) If John knows this sentence, he certainly knows what Mary's phone number is. So the term $[4,4,2,4,2,8,0]$ is an id constant for Mary's phone number; and more generally, a list of seven digits forms an id constant for a phone number. We cannot express these ideas in Konolige's logic. First of all, the ground term $[4,4,2,4,2,8,0]$ is not a constant, so it cannot be an id constant. More importantly, the logic does not allow us to describe the naming map. The constants of the representation language are not included in the domains of discourse of the structures for this logic. No variable can range over constants of the representation language. The only way to refer to a constant is by writing it explicitly in the scope of the belief operator. We cannot represent the statement "a list of digits is an id constant for a phone number", because it is a general statement about an infinite set of id constants. Konolige considered an extension of his logic that would address this problem, but he did not incorporate this extension in the final version – the one for which he proved soundness and completeness.

Now consider a robot who uses an internal clock, which counts the time in seconds since the robot's CPU was last booted. Let $internal(n)$ be the interval when the internal clock reads n seconds. Suppose the robot wants to get to a meeting on time, and someone tells him "The meeting starts in 5 minutes". The robot knows that he heard this statement during the interval $internal(1000)$, and he knows that it will take 2 minutes to get to the meeting. He infers that he should leave during the interval $internal(1180)$. Then the robot knows how to get to the meeting on time: he will wait until his internal clock reads 1180, and then leave. In this situation we could reasonably say that the robot knows when the meeting will start. So it seems that $internal(1180)$ is an id constant for an interval of time.

On the other hand, suppose someone tells the robot "The meeting will start at 9:30 AM". Let $time(h,m)$ denote the one-minute interval that begins h hours and m minutes after midnight (ignoring the date for brevity). Then the term $time(9,30)$ denotes an interval containing the time when the meeting starts. In this situation also, we could reasonably say that the robot knows when the meeting will start. So it seems that $time(9,30)$ is an id constant for an interval of time. It is quite possible that the robot still does not know at what time on his internal clock the meeting will start. If not, he does not yet know how to get to the meeting on time. Suppose that during the interval $internal(1000)$, the robot sees a clock that reads 9:25. Then the robot can infer that the meeting will start at $internal(1300)$, and he knows how to get to the meeting.

In this case it seems that there are two naming maps. There is a public system for naming times, using hours and minutes, and we can reasonably say that the robot knows when the meeting will start if he knows that is at 9:30 AM. So we need a naming map that takes an interval of the form $time(h,m)$ and maps it to the name $time(i,j)$, where i and j are numerals for h and m. The robot also has an internal clock, and we can say that the robot knows when the meeting starts if he knows that is at $internal(1300)$ – even if he does not know what clock time corresponds to $internal(1300)$. Then we need another naming map, which takes an interval of the form $internal(n)$ and maps it to the name $internal(i)$, where i is a numeral for n. We would like to be able to define these two naming maps, and to specify which one we are using in each particular case. In Konolige's logic, it is not possible to have more than one naming map, nor is it possible for the user to define a new naming map.

1.2 Defining and Using Naming Maps

The range of a naming map is a set of names – typically an infinite set. If we are to write an axiom that defines a naming map, we must be able to quantify over names. If the representation language is first-order logic, names are ground terms. Let us consider an epistemic logic in which ground terms of the representation language are members of the domain of discourse. I will present this logic informally, and show that we can use it to define the naming maps that arise in our examples.

Let $\mathbf{M}$ be a structure for our language. Its domain includes the ground terms of the representation language, and it assigns to each agent the set of sentences that he believes. Let s be an assignment of values to the variables of our language. Let t be a term containing free variables $x_1, ..., x_n$, and suppose that $s(x_1), ..., s(x_n)$ are all ground terms of the representation language. Then the denotation of the term $`t$ in $\mathbf{M}$ and s is $t[s(x_1)/x_1, ..., s(x_n)/x_n]$ – the term formed by simultaneously substituting $s(x_1), .., s(x_n)$ for $x_1, ..., x_n$ in t. Since $x_1, ..., x_n$ are all the free variables of t, and $s(x_1), ..., s(x_n)$ are ground terms, it is clear that $t[s(x_1)/x_1, ..., s(x_n)/x_n]$ is a ground term. The symbol ` serves as a quotation mark, but free variables under the quotation mark are not quoted; instead we replace them with their values.

We can use this notation to describe the naming map for telephone numbers. For each integer from 0 to 9 there is a digit, and we can define this correspondence by a simple list of axioms:

[3] $digit(0,`0) \wedge digit(1,`1) \wedge ... \wedge digit(9,`9)$

We now introduce the predicate "$ph\_id(n,x)$", which means that n is a phone number and x is the name as-

signed to n by the naming map for telephone numbers. We represent lists using "cons" and "nil".

[4] $(\forall n, d. digit(n, d) \rightarrow ph\_id(cons(n, nil), `cons(d, nil)))$
[5] $(\forall n, d, l, k.\ (digit(n, d) \wedge ph\_id(l, k)) \rightarrow ph\_id(cons(n, l), `cons(d, k)))$

Axiom (4) tells us that since the constant 5 is the digit for the number 5, the naming map for phone numbers assigns the term $cons(5, nil)$ as a name for the list $cons(5, nil)$. Axiom (5) handles the recursive case.

This notation allows us to define a naming map; we need further notation so that we can use the naming map to describe an agent's beliefs. Once more, let **M** be a structure for our language, and s an assignment of values to variables. Suppose t is a term that denotes an agent a, and A is a wff with free variables $x_1, ..., x_n$. Assume that $s(x_1), ..., s(x_n)$ are ground terms of the representation language. Then $bel(t, A)$ is a wff, and it is true in **M** and s iff the sentence $A[s(x_1)/x_1, ..., s(x_n)/x_n]$ is one the beliefs that **M** assigns to agent a.

Suppose John believes the sentence $phone(mary, n)$, where n is the name that the naming map for phone numbers assigns to Mary's phone number. We can assert this as follows:

[6] $(\exists n, k.\ phone(mary, k) \wedge ph\_id(k, n) \wedge bel(john, phone(mary, n)))$

Now assume that Mary's phone number is 5766, and John believes that Mary's phone number is 5766:

[7] $phone(mary, cons(5, cons(7, cons(6, cons(6, nil)))))$
[8] $bel(john, phone(mary, cons(5, cons(7, cons(6, cons(6, nil))))))$

The new logic will allow us to prove that (7),(8), and (3)-(5) entail (6). This example show how a user can define a naming map and prove that the map assigns a particular name to a particular object. It also shows how one can use a naming map to describe an agent's beliefs.

To define the naming maps for intervals of time, we must first describe a naming map from integers to numerals. We will settle for the simplest possible kind of numerals, using the constant 0 and the successor function. These are woefully inefficient; defining a more practical system of numerals is left as an exercise for the reader. $numeral(n, i)$ means that i is a numeral for the integer n. We have:

[9] $numeral(0, `0)$
[10] $(\forall n, i. numeral(n, i) \rightarrow numeral(s(n), `s(i)))$

Given this naming map, we can easily define two naming maps for times. The domain of the first naming map is the set of all intervals of the form $internal(n)$. Let $name1(i, j)$ mean that i is an interval of the form $internal(n)$, and j is the name assigned to i by the first naming map. We define this map by the following axiom.

[11] $(\forall i, j.\ numeral(i, j) \rightarrow name1(internal(i), `internal(j)))$

Let $name2(i, j)$ mean that i is an interval of the form $time(h, m)$, and j is the name assigned to i by the second naming map. We define this map by the following axiom.

[12] $(\forall h, j, m, k.\ (numeral(h, j) \wedge numeral(m, k)) \rightarrow name2(time(h, m), `time(j, k)))$

If we base our epistemic logic on a sentential theory, the user of the logic must be able to define new naming maps. If users are limited to pre-defined naming maps, they will never be able to handle quantification into the scope of attitudes. The range of naming map is usually an infinite set of names. Then in order to define a naming map, we must be able to quantify over names. I have given an informal sketch of a logic that allows quantification over names. Section 1.3 will describe a technique for proving theorems in the logic, by translating it to first-order logic. The logic suffers from the problem of logical omniscience: each agent believes every logical consequence of his beliefs. Section 2 describes an extension of the logic that allows a distinction between what is believed and what logically follows from the beliefs. Section 3 describes an implementation: an incomplete but useful theorem-prover that works by translating a subset of the logic into Prolog.

1.3 Translation to First-Order Logic

We are given sentences A and B in some logical language L, and we wish to show that A entails B. A standard approach is to show that $(A \wedge \neg B)$ is unsatisfiable (has no model). Suppose we have an algorithm that takes a sentence C in L and returns a sentence C' in first order logic such that C' has a model iff C has a model. C' is called the *translation* of C. If we apply this algorithm to the sentence $(A \wedge \neg B)$, and show that the resulting first order sentence is unsatisfiable, we have shown that A entails B. This technique reduces the problem of theorem proving in a new language to the problem of theorem proving in first order logic. Given the progress that computer scientists have made in first order theorem proving, such a reduction has obvious advantages. Of course, everything depends on the nature of the translations. If they are much larger than the original sentences, or their logical structure creates difficulty for our first order theorem provers, then it might be better to do our theorem proving in the original language.

Moore (1980) proposed a simple technique for translating from a modal logic of knowledge to first order logic. The modal logic assigns to each agent a set of possible worlds: the worlds that are compatible with that the agent's knowledge. The agent knows a sentence A iff A is true in each of these worlds. Consider the following sentence of modal logic:

[13] $[S](\forall x.P(x) \vee Q(x))$

This says that agent S knows that everything is either a P or a Q. That is, for every world w that is compatible with S's knowledge, everything is either a P in world w or it is a Q in world w.

Suppose that for each predicate letter P of rank n in the modal language, there is a predicate letter $:P$ of rank $n+1$ in the corresponding first order language. $:P(w, x_1, ..., x_n)$ means that the predicate P holds for the individuals $x_1, ..., x_n$ in the possible world w. Suppose K is a predicate of the first order language and $K(a, w)$ means that world w is compatible with the knowledge of agent a. Using the new predicates, we would translate the modal sentence (13) into the following first order sentence.

[14] $(\forall w.K(S, w) \rightarrow (\forall x. : P(w, x) \vee : Q(w, x)))$

This example is simplified in several ways. For example, in Moore's system an agent's knowledge can vary from world to world, so the predicate K needs three arguments, not two. However, the example shows the crucial properties of Moore's translation. The wff that appears as the argument of the modal operator in (13) re-appears in a recognizable form in the first order sentence (14). The quantifier and the connective are the same as before, but each predicate letter has an extra argument. The modal operator is transformed into a quantifier over possible worlds, with a range restriction to indicate the agent.

I propose to translate the sentential formalism into first order logic in much the same way that Moore translated a possible worlds formalism into first order logic. Where Moore uses the set of all possible worlds that are compatible with the agent's knowledge, I will use a set S of first order models of the agent's belief set Γ. In Moore's logic, a sentence is known iff it is true in all worlds compatible with the agent's knowledge. Given that the agent's belief set Γ is deductively closed, it is easy to see that a sentence is believed iff it is true in all models of Γ. If a sentence A is believed then $A \in \Gamma$, so A is true in all models of Γ. Suppose A is not a believed. Then since the beliefs are deductively closed, A is not provable from the beliefs. Then by completeness of first order logic, there exists a model of $\Gamma \cup \{\neg A\}$, so A is not true in every model of Γ.

The most obvious idea is to use the set of all models of the belief set Γ. This is impossible, because in general there is no such set: the models of Γ may form a proper class. There is another reason for not using all the models of Γ. The standard definition of truth in first order (or modal) logic relies on a recursive definition of the notion of an assignment satisfying a wff. We need satisfaction, not just truth, to describe the semantics of quantifiers. However, it is well known that if every individual in the domain of the model is the denotation of some ground term, we can do without the notion of satisfaction and thereby simplify the semantics (Gallier 1986, p. 163).

Let $\mathbf{M} = (M, I)$ be a structure for a language L; we say that $\mathbf{M}$ is an *eponymous* structure iff for every $x \in M$ there is a $t \in ground(L)$ such that x is the denotation of t in $\mathbf{M}$. Let D be an arbitrary countably infinite set. A *D-L-structure* is an eponymous L-structure $\mathbf{M}=(M, I)$ such that $M \subseteq D$. Note that the D-L-structures form a set, not a proper class. If a D-L-structure is a model of Γ, we say that it is a *D-L-model* of Γ. We have already observed that if a set of sentences Γ is deductively closed, then $A \in \Gamma$ iff $\mathbf{M} \models A$ for every model $\mathbf{M}$ of Γ. We can prove a similar result for eponymous models.

Lemma 2.1. Let Γ be a deductively closed set of sentences in a first-order language LB. Let LB' be a language formed by adding a countable infinity of new constants $\{c_1, c_2, c_3, ...\}$ to LB. Then Γ is equal to the set of sentences that hold in every D-LB'-model of Γ.

Proof: Suppose $A \in \Gamma$; then if $\mathbf{M}$ is a D-LB'-model of Γ, $\mathbf{M} \models A$. Suppose $A \notin \Gamma$; we must show that there is a D-LB'-model of Γ in which A is false. Since Γ is deductively closed, $\neg A$ is consistent with Γ, and by completeness of first-order logic there is an LB-structure that is a model of $\Gamma \cup \{\neg A\}$. By the Löwenheim-Skolem theorem, there is an LB-structure $\mathbf{M}_0$ with a countable domain that is a model of $\Gamma \cup \{\neg A\}$. Since D is countably infinite, there exists a 1-1 onto mapping between a subset M_1 of D and the domain of $\mathbf{M}_0$. Use this mapping to construct an LB-structure $\mathbf{M}_1 = (M_1, I_1)$ that is isomorphic to $\mathbf{M}_0$; then $\mathbf{M}_1$ is a model of $\Gamma \cup \{\neg A\}$. Choose a function f from the set $\{c_1, c_2, c_3, ...\}$ onto M_1, and consider the LB' structure $\mathbf{M}_2 = (M_1, I_2)$, where $I_2(g)$ is defined as follows: $I_2(g) = I_1(g)$ for every function letter and predicate letter g in LB, and $I_2(c) = f(c)$ for every $c \in \{c_1, c_2, c_3, ...\}$. Clearly $\mathbf{M}_2$ is an eponymous structure, and $M_1 \subseteq D$, so $\mathbf{M}_2$ is a D-LB'-structure. $\mathbf{M}_1 \models (\Gamma \cup \{\neg A\})$, so $\mathbf{M}_2 \models (\Gamma \cup \{\neg A\})$. Then $\mathbf{M}_2$ is a D-LB'-model of Γ in which A is false. □

This lemma tells us that if the set of beliefs is deductively closed, then A is a belief iff A is true in every D-LB'-model of the beliefs. Then we can use D-LB'-models of the agent's beliefs in much the same way that Moore used possible worlds compatible with the agent's knowledge. If $\mathbf{M}$ is a D-LB'-model, the sentence $(\forall x.A)$ is true in $\mathbf{M}$ iff for every $t \in ground(LB')$, $A[t/x]$ is true in $\mathbf{M}$. This means that we can avoid the notion of satisfaction and use truth instead. This simplifies the problem a good deal.

Consider the belief sentence $pr(j, (\forall x.P(x)))$. The above discussion suggests that its translation might be as follows:

[15] $(\forall m.model(m, j) \rightarrow (\forall x.gr(x) \rightarrow !P(x, m)))$

Here $model(m, j)$ means that m is a D-LB' model of John's beliefs, $gr(x)$ means that x is a ground term of LB', and $!P(x, m)$ means that the atomic sentence $P(x)$ is true in the structure m. The proposed translation says that if m is a D-LB' model of John's beliefs and

$x \in ground(LB')$, then $P(x)$ is true in m. By the argument above, this is true iff John believes $(\forall x.P(x))$.

However, this kind of translation would lead to a very expensive theorem-prover. Notice that a universal quantifier in a belief translates to a universal quantifier with an extra range restriction. If A translates to A', then $(\forall x.A)$ translates to $(\forall x.gr(x) \rightarrow A')$. Suppose that we use a resolution theorem-prover, and John believes a clause containing n universal variables. The translation of this clause would have to contain n new literals – one range restriction for each of the n universal variables. A theorem-prover that had to resolve away all these extra literals could never hope to match the efficiency of Konolige's theorem-prover.

The first-order sentence (15) is an obvious translation of the belief sentence $pr(j, (\forall x.P(x)))$, but it is inefficient because it introduces a new literal for each quantifier under a belief operator. The translation I propose is not (15), but the following sentence.

[16] $(\forall m.model(m, j) \rightarrow (\forall x.!P(x, m)))$

The variable x ranges over the whole domain, including objects that are not ground terms of LB'. In order to get the desired truth conditions, we define the semantics of the predicate letter $!P$ so that if its first argument is not a ground term of LB', it coerces that argument to a ground term of LB'. This coercion is loosely similar to coercion in a programming language. When the programmer writes "5 + 3.14159", the machine coerces the fixed point number 5 into a floating point number before applying floating-point addition. The predicate letter $!P$ will coerce its first argument to a ground term of LB' by applying a function that maps the domain onto the set $ground(LB')$.

The full version of this paper (Haas 1993) demonstrates that proof by translation is sound and complete. The argument is in three parts. First we define a relation between belief structures and first-order structures called *correspondence*, and we show that if $\mathbf{M}_1$ and $\mathbf{M}_2$ are corresponding structures, then A is true in $\mathbf{M}_1$ iff the translation of A is true $\mathbf{M}_2$. Next we show that if A is satisfiable, there exists a belief model $\mathbf{M}_1$ of A and a first-order structure $\mathbf{M}_2$ such that $\mathbf{M}_1$ and $\mathbf{M}_2$ are corresponding structures. $\mathbf{M}_2$ is a model of the translation of A, so the translation of A is satisfiable. This implies that the proof technique is sound: if we use a sound first-order proof system to show that the translation of A is unsatisfiable, then A itself must be unsatisfiable. Next we show that if the translation of A is satisfiable, there exists a model $\mathbf{M}_2$ of the translation of A and a belief structure $\mathbf{M}_1$ such that $\mathbf{M}_1$ and $\mathbf{M}_2$ are corresponding structures. $\mathbf{M}_1$ is a belief model of A, so A is satisfiable. This implies that the proof technique is complete: if A is unsatisfiable, its translation is also unsatisfiable, and by using a complete first-order proof system we can demonstrate this unsatisfiability.

2 Limited Inference

2.1 Creary's Proposal for Limited Inference

People and robots infer new beliefs from old ones, but they certainly cannot infer everything that follows from their beliefs. Theories of attitudes based on possible worlds are unable to explain this fact. They predict that every agent believes every logical consequence of his beliefs (Moore 1985). If an agent cannot infer every logical consequence of his beliefs, we say that the agent's inference is limited. Sentential theories are clearly capable of describing limited inference. Since the beliefs are sentences, the theorist is free to choose an algorithm for manipulating those sentences, and to assert that an agent uses that algorithm to create new beliefs from old ones. Of course, this does not mean that it is easy to find an algorithm that captures the strengths and limitations of human reasoning. The sentential theories allow us to tackle this problem. No theory, sentential or otherwise, comes close to solving it.

Our translation uses models of the agent's beliefs in much the same way that modal logic uses possible worlds. As a result, the epistemic logic of section 1 describes agents who are logically omniscient. For each agent A, the set A's beliefs is deductively closed. The topic of this paper is naming maps, not limited inference, and I will not develop a full treatment of limited inference. However, it is important to show that the translation technique does not force us to accept logical omniscience. That would mean giving up a major advantage of sentential theories. The full version of this paper (Haas 1993) describes an extended epistemic logic, with a new epistemic operator *bel*. The atomic sentence $bel(A, P)$ means that agent A believes P, while the sentence $pr(A, P)$ means that P logically follows from A's beliefs. Therefore $pr(A, P)$ holds iff there exist sentences $P_1, ..., P_n$ such that $bel(A, P_1)$,...,$bel(A, P_n)$ all hold, and P logically follows from $P_1, ..., P_n$. So $bel(A, P)$ entails $pr(A, P)$, but the converse is false. The logic does not require that beliefs are closed in any way. For example, it is consistent to assert $bel(A, P \wedge Q) \wedge \neg\, bel(A, Q \wedge P)$. So this logic does not require that everything provable is believed – indeed, it does not require that anything provable is believed. It simply leaves open the question of which of the provable sentences will actually be proved, and so believed. The full paper describes a translation from this logic to first-order logic, and proves soundness and completeness.

Suppose P entails Q. Then in the extended logic, $bel(A, P)$ entails $pr(A, Q)$. If we wish to prove $bel(A, Q)$, we need some principle which asserts that under certain conditions, what is provable is also believed. Creary (1979) proposed such a principle. Suppose agent A has a description of the beliefs of agent B, stated in the extended logic. Suppose A uses this description to prove

$pr(B, P)$. Then A knows that B's beliefs entail P. But A knows more: he can learn by introspection that he himself has proved, from a description of B's beliefs, that those beliefs entail P. Suppose A thinks that whatever A can prove, B can also prove (given the same premisses). Roughly speaking, A thinks that B is at least as smart as A is. Now B has direct access to his own beliefs, while A has only a description of B's beliefs. It should harder to prove $pr(B, P)$, given a description of B's beliefs, than to prove P, given the beliefs themselves. Then A can argue as follows: "I was able to prove $pr(B, P)$ from a description of B's beliefs. It is harder to prove this than it is to prove P from the beliefs themselves. Therefore B will be able to prove P from his beliefs." A can even estimate the time that B needs to prove P: it is probably no more than the time that A needed to prove $pr(B, P)$. In this way agent A can predict B's inferences without having a detailed theory about the inference algorithm that B uses. A relies on empirical observations of the behavior of his own inference algorithm, together with a single assumption about B's inference algorithm: it is at least as effective as A's.

To make this reasoning sound, we must be careful about what we mean by "a description of B's beliefs". In particular, we must be certain that this "description" does not include sentences of the form $pr(B, P)$. Agent B has direct access to his own beliefs, but he has no direct access to the sentences that are provable from his beliefs. If A has information about what is provable from B's beliefs, he has information about B's beliefs that may well be unknown to B himself. Then it may be easier for A to prove $pr(B, P)$ than for B to prove P. For example, suppose A knows that B believes Peano's axioms, and he has been told that Peano's axioms imply the Chinese Remainder Theorem. Let the sentence CRT express the Chinese Remainder Theorem. Then A knows that $pr(B, CRT)$, and he can trivially prove $pr(B, CRT)$. This certainly does not imply that if B is as smart as A, B can prove CRT. A's proof relies on information about B's beliefs that may well be unknown to B, so we cannot argue that it is harder for A to prove $pr(B, CRT)$ than it is for B to prove CRT.

We can avoid this problem by a simple restriction: a description of B's beliefs may not contain any occurrence of the epistemic operator pr. That is, a description of B's beliefs may use only the operator bel. Since this operator describes actual beliefs of B, which are all directly accessible to B, it seems plausible that a description using this operator contains no extra information about B's beliefs – no information that might be unknown to B. Notice that under this definition, a description of B's beliefs may contain all kinds of irrelevant information – information about the beliefs of other agents, or information that has nothing to do with anybody's beliefs. This is harmless, since A cannot use this irrelevant information to prove sentences of the form $pr(B, P)$.

Agent A, after proving P from the premisses $S_1, ..., S_n$, must be able to observe this event and form a belief that describes it. In order to express this belief, we need a new epistemic operator. The sentence

[17] $prove(A, [S_1, ..., S_n], P, t)$

means that A proved sentence P from the premisses $S_1, ..., S_n$, and it took time t to find this proof. Agent A forms this belief by observing an event, in much the same way that he might observe a leaf falling from a tree. The main difference is that since this event took place inside the agent's head, it is easier to observe than an event in the external world.

Agent A must now apply the principle "If I can prove $pr(B, P)$ from premisses $S_1, ..., S_n$ in time t, and the operator pr does not occur in $S_1, ..., S_n$, then B will also be able to prove P in time t." In order to apply this principle, we must add a time argument to our belief and provability operators. We cannot state the principle as a single sentence of our logic, because P and $S_1, ..., S_n$ are universal variables ranging over sentences of the representation language. Our logic allows quantification over ground terms only – not over sentences. We can state the principle as an axiom schema, however. For any sentences P and $S_1, ..., S_n$ of our language, with no occurrence of pr in $S_1, ..., S_n$, and any constant B denoting an agent, A has the axiom

[18] $(S_1 \wedge .... \wedge S_n \wedge prove(A, [S_1, ..., S_n], pr(B, P, t_0), t)) \rightarrow bel(B, P, t_0 + t)$

The left side of the implication says that $S_1, ..., S_n$ correctly describe B's beliefs, and A has proved from $S_1, ..., S_n$ that P follows from the sentences that B believes at t_0. Since A took t units of time to prove this, B will need at most t units of time to find a proof of P, so B will actually believe P at time $t_0 + t$ – that is, t units after t_0.

3 Implementation

Having proved a completeness theorem, one should not assume that the next task is to implement a complete theorem prover. It is possible that an incomplete but efficient theorem prover will prove more useful. The classic example is of course the incomplete theorem prover used in Prolog. Let us consider a Prolog-style implementation for the new epistemic logic.

Suppose that we extend the logic used in Prolog by including the quotation mark and the attitude operator pr. We restrict the argument of pr to be a definite clause. The effect of this is to ensure that the agents' beliefs consist entirely of definite clauses. Let us include explicit universal quantifiers, in order to make it clear when a variable is bound inside the scope of the attitude operator. If the variable is bound outside the scope of the

operator, we omit the quantifier. In this language we can write programs like the following:

```
[36]
pr(john,(∀ N. phone(mary,N) :- phone(william,N))).
pr(john,
   phone(william,cons(5,cons(7,cons(6,cons(6,nil)))))).
ph_id(cons(N,L),`cons(D,K)) :- digit(N,D),ph_id(L,K))))).
ph_id(cons(N,nil),`cons(D,nil)) :- digit(N,D))).
digit(0,`0).
...
digit(9,`9).
```

Given this program, the goal

[37] :- pr(john,phone(mary,N)),ph_id(K,N).

should succeed with N=`cons(5, cons(7, cons(6, cons(6, nil)))). That is, we should be able to prove that John believes that Mary's phone number is N, where N is a list of digits, by first showing that John believes that Mary's number is 5766, and then showing that 5766 is a list of digits.

Suppose that a belief atom appears on the left side of Prolog's :- operator, as in the clause

[38] pr(a,(p :- q)) :- r.

Translating to first-order logic, we get a definite clause:

[39] !p(M) :- model(a,M),!q(M),r.

Now suppose a belief atom appears on the right side of the :- operator, as in the clause

[40] r :- pr(a,(p :- q)).

In this case, we do not get a definite clause. Instead, we get a formula with an implication on the right side:

[41] r :- (∀ M. !p(M) :- model(a,M),!q(M)).

Skolemizing, we get:

[42] r :- (!p(skolem0) :- model(a,skolem0),!q(skolem0)).

This wff is a *hereditary Harrop formula* (Miller 1989), and it is easy to see that if we take a formula in the given subset of our epistemic logic, and translate to first-order logic, we will always get a hereditary Harrop formula. Hereditary Harrop formulas are a generalization of definite clauses, because they allow implications to appear on the right side of a rule.

Miller (1989) proposed a simple proof rule for these implications. In order to prove (p :- q) from a set of clauses S, we prove p from the set $S \cup \{ q \}$. This proof rule is not complete, but it is strong enough to be quite useful. We can implement this proof rule by translating hereditary Harrop formulas into definite clauses. The idea is to add to each predicate an extra argument: a list of assumptions. In order to prove (p :- q), we add q to the current list of assumptions, and then prove p. If we need to prove r from the current list of assumptions L, and r is a member of L, we succeed at once. I have used this simple technique to build a theorem-prover for a subset of the epistemic logic, which works by first translating from epistemic logic to (a subset of) hereditary Harrop formulas, and then compiling the Harrop formulas into Prolog clauses. The translation must also include the extra axioms, as given in the definition of translation above. The reader can check that these axioms are all expressible as definite clauses.

4 Conclusions and Future Work

Quantification into the scope of attitudes is arguably the hardest problem in the study of propositional attitudes. It seems clear that to analyze this phenomenon in a sentential theory, we need a mapping from objects to their names in the representation language. The philosophical work of the last fifteen years (especially (Boër and Lycan 1986)) has established that there is no universal mapping from objects to names – an object can have more than one useful name, and which one we want depends on our goals. Haas (1986) expressed similar ideas, without offering any proposals for implementation. In the meantime, AI workers have been trying to find resolution theorem provers for epistemic logics. The logics they work on include only trivial treatments of quantification into the scope of attitudes. Konolige simply postulates a class of distinguished constants. His notation cannot express any theory about these constants, nor is there any theory built into the formalism. I have proposed a logic that allows the user to define the naming maps that are needed for particular domains. This logic does not solve the problem, but it allows us to express the kind of knowledge that is needed for a solution. It also allows for a resolution theorem-proving algorithm, and even for a straightforward (though incomplete) Prolog implementation.

The most important shortcoming of the new logic is that an attitude operator cannot appear in the scope of an attitude operator. That is, there are no beliefs about beliefs. It is straightforward to add nested beliefs to the definition of our language; the difficulty is to extend the translation to handle them. The first step is to allow repeated application of the exclamation point: if !c denotes the constant c, then !!c denotes the constant !c, and so on. We must then apply the translation process recursively. To translate bel(john,P) we first translate P to an orthodox sentence P'. Then we can translate bel(john,P') to a first-order sentence, using the technique we already have. The only real difficulty arises when P contains free variables; then P describes a set of sentences, and P' must describe the set of translations of these sentences.

I believe this is possible, but it requires translation to a language that is slightly more expressive than first-order logic, and this in turn requires a special unification algorithm.

Acknowledgement. This work was supported by the National Science Foundation under grant number IRI-9006832.

5 References

Boër, Steven E., and Lycan, William G. 1986. Knowing Who. MIT Press: Cambridge, Massachusetts.

Creary, L. G. 1979. Propositional Attitudes: Fregean Representation and Simulative Reasoning. in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence.* Tokyo, Japan: 176-181.

Gallier, Jean H. 1986. Logic for Computer Science. Harper and Row: New York.

Haas, Andrew. 1986. A Syntactic Theory of Belief and Action. *Artifical Intelligence* 28(3), 245-292.

Haas, Andrew. 1993. An Epistemic Logic with Quantification over Names. submitted to *Computational Intelligence.*

Kaplan, David. 1975. Quantifying In. In Donald Davidson and Gilbert Harman, Eds., *The Logic of Grammar.* Dickinson: Encino, California: 112-144.

Konolige, Kurt. 1986a. A Deduction Model of Belief. Pitman: London.

Konolige, Kurt. 1986b. A Resolution Method for Quantified Modal Logic. in *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference.* Joseph Y. Halpern, ed. Morgan Kaufmann: Los Altos, California: 309-324.

McCarthy, John. 1979. First-Order Theories of Individual Concepts and Propositions. in B. Meltzer and D. Michie (eds.), *Machine Intelligence 9.* Halsted Press: New York: 120-147.

Miller, Dale. 1989. A Logical Analysis of Modules in Logic Programming. *Journal of Logic Programming,* 6(1), 79-108.

Moore, Robert C. 1980. Reasoning about Knowledge and Action. Technical Report 191, SRI International, Menlo Park, California.

Moore, Robert C., and Hendrix, Gary G. 1982. Computational Models of Belief and Semantics of Belief Sentences. in *Processes, Beliefs, and Questions.* S. Peters and E. Saarinen, eds. Dordrecht: D. Reidel: 107-127.

Morgenstern, Leora. 1987. Knowledge Preconditions for Actions and Plans. in *Proceedings of the Tenth International Conference on Artificial Intelligence.* Morgan Kaufmann: Los Altos, California: 867-874.