KNOWING THAT AND KNOWING WHAT

Allan Ramsay

1987

Cognitive Science Research Paper

Serial No. CSRP. 074

Cogntive Studies Programme
The University of Sussex
Brighton BN1 9QN

CARNEGIE-MELLON UNIVERSITY
PITTSBURGH, PENNSYLVANIA 15213

074

KNOWING THAT AND KNOWING WHAT

Allan Ramsay

Cognitive Studies Programme
University of Sussex. Falmer BN1 9QN

ABSTRACT

This paper presents a set of inference rules for reasoning about other people's knowledge. These rules are not complete, but they are effective. The paper shows how they may be used with a non-trivial set of contingent facts and rules about the knowledge possessed by various people to answer fairly complex queries about who knows what. The inference rules are based on the notion of mimicking the reasoning that might be performed by another person, rather than on an analysis of the conditions under which some fact is constrained by the other facts which may be available. The conclusion argues that the speed of the resulting system at least partially compensates for the fact that it does not manage to derive every possible conclusion.

INTRODUCTION

Consider the following situation: Allan knows that Alex and Trish live together. He also knows that Janet knows Alex's phone number. How does he know that asking Janet what Alex's number is will enable him to phone Trish?

This problem is typical of the sort of reasoning we have to perform every day in order to function in the world. We need to be able to reason not just with our own knowledge about the world, but with other people's. We need it in order to be able to work co-operatively with other people, since we have to be able to predict what information they will need in order to carry out their part of the task. We need it to be able to work against them, so that we can set them traps (if, for instance, you were playing someone at chess you would not try to catch them with a fool's mate if you knew they already knew about it). We need it in order to be able to carry out satisfactory dialogues with other people - if you assume they know more than they in fact do then they will fail to understand you, if you assume that they know less then they will get bored.

Reasoning about other people's knowledge is recognised in AI as a major task. A number of people have proposed ways of dealing with it, notably Moore (1984) and Konolige (1982). These solutions tend to be elegant but computationally intractable. Moore (1980), for instance, explicitly acknowledges that at that time he had no working theorem prover for

Knowing that and knowing what

This paper presents an alternative scheme for reasoning about other people's knowledge. The mechanism described below is incomplete, in that there may well be things which follow from what you know about another person's knowledge but which this system cannot infer. It is, however, fast enough to be included within, for instance, a natural language generation system without completely swamping the processing done by the rest of the system. The present implementation contains rules for analysing problems like the one that introduced this section of the paper.

OBSERVATIONS ON REASONING ABOUT KNOWLEDGE

The following points are critical for any attempt to reason about other people's knowledge (these points are well-known, and I claim no credit for pointing them out here. Nonetheless they need to be restated).

(i) No-one can know anything that is not true. As a direct consequence, if A knows that B knows P, then A must know P. Knowledge thus differs from belief, since there is no reason to suppose that if A believes that B believes P then A will also believe it. The distinction is slightly artificial, since no-one ever really *knows* anything, except perhaps logical tautologies. All we ever have is very strong beliefs. Nonetheless, people and AI planning systems frequently behave as though they thought they did know things. The task of recovering when what was thought to be known turns out to be wrong is, of course, another open problem in AI. We will follow all other AI workers on knowledge about knowledge by assuming that someone else will solve the truth maintenance problem for us. The current system assumes that statements about its own and other people's knowledge are indeed correct, and are not just beliefs.

(ii) In general, then, knowing that someone else knows something is grounds for inferring it for yourself. There is one very specific case in which it fails. If I overhear Janet saying "It's raining", then I can infer that it is indeed raining (subject to all the usual restrictions about the correctness of Janet's belief). From this I can, if I want, infer that I know it's raining, or do whatever else I want with it. If I overhear her saying "I know Alex's phone number" then all I can infer is that Alex has a phone number. In case (i), I know exactly what Janet knows as soon as I know she knows it. In case (ii) I know less than she does.

(iii) The mechanism described in (i) is transitive - if A knows that B knows that C knows that P then A can infer P. This chaining of knowledge about people's knowledge can be carried out as many times as you like.

(iv) It is generally assumed that everyone is capable of performing the same sets of inferences. This is yet another idealisation, though it is not usually pointed out with as much care as the others that are made in this area. If we can assume this, then much of the difficulty of dealing with knowledge about knowledge disappears. In order to see whether someone else is capable of drawing some conclusion all we need to do is see how we would draw it ourselves, and then check that we have not used any contingent facts or rules which we do not know are available to them.

This approach can be extended to deal with people whose ability to reason is inferior to our own. simply by regarding some or all of our rules of inference as contingent rules to which they do not have access. It cannot be extended to people whose reasoning is

either R1:(P1 & ... &Pn) → Q1 or R2: (P1 & ... & Pn) → Q2. Then if A knows that B knows P1 and P2 and ... Pn, then he knows that B knows Q1 or Q2, and hence can infer for himself that Q1 OR Q2 is true. But this is different from, and less than, what B knows - A just knows that Q1 OR Q2 is true, whereas B really does either know that Q1 is true or know that Q2 is). There is a final case we might need to consider when worrying about other people's ability to reason, namely when their reasoning is not better or worse than our own, but different in some other way. It might be that their reasoning differs from ours by employing a different, but equally valid, set of axioms and rules of inference. If we knew this, and knew what they were, we could again take their axioms and rules to be contingent facts which we knew they had access to, and continue to mimic them. Since we are concerned with knowledge and not belief, there is no other way for their reasoning to differ from our own. If we know that they use a sound inference rule, and if furthermore we know what it is, then we can use it ourselves.

A consequence of the above discussion is that we can examine other people's knowledge simply by seeing what we ourselves can infer, and then checking we know that they know all the basic facts we have used. The remainder of the paper presents a set of inference rules, and a collection of contingent facts and rules about various people's knowledge, which will support the kind of reasoning needed for the introductory example. The inference rules are not complete. In particular, they do not support reasoning from hypotheses. It should be remembered that any set of inference rules must either be incomplete or run the risk of non-termination. The rules given below could easily be extended to cover extra cases as required.

RULES FOR REASONING ABOUT KNOWLEDGE

The following rules are written in a PROLOG-like notation, with variables indicated by a preceding @ sign, and the heads and bodies of clauses separated by "if". It is important to note that the inference engine used stops mutually recursive clauses from producing infinite loops (so that it is quite safe, for instance, to use rules like "X lives with Y if Y lives with X"). Full details and program text for this inference engine are given in Ramsay and Barrett (1987).

The rules are split into two sections. The first group are general rules of inference. The second group are contingent facts and rules, which are marked as being known by one or more people. The system uses the first group on the second to draw conclusions about various people's knowledge. It must be clearly understood that the reasoning system we are interested in consists of the inference rules plus the contingent facts, and that no contingent fact can be provided without the system knowing it - it just would not make sense.

Inference rules The following rules are used by the system for its own inferences. It is assumed, however, that the same set of rules are available to everyone else, so that anything that can be concluded using these rules and the facts available to some specific other person can be inferred by that person.

The rules make use of three conventions, as follow:

(i) an expression such as **knows(a, f, [b,c,d])** should be read as "D knows that C knows that B knows that A knows that F". In other words, the third argument to **knows** is used for nesting statements about what people know about each other's knowledge. This way of expressing such statements enables us to delay worrying about whether people really do know what we have hypothesised them to know until we need to check the particular facts they have access to. The effect could have been achieved by some rule which converted a nested expression to the form we want, for instance

knows(@A, knows(@B, @P)) if knows1(@B, @P, [@A]).

We leave the rule in the form which we actually want it in for simplicity.

(ii) Contingent facts and rules are represented by statements of the form

fact(@N, @X, @P)

where **N** is an index for the fact (so that it may be referred to elsewhere), **X** indicates who knows it, and **P** is the fact itself. For rules, **P** is a compound expression containing an implication arrow. Thus

fact(1, Janet, that(raining))

says that the first fact the system knows about is that Janet knows it is raining (we sometimes say below "'it is raining' is a fact for Janet"). The indices are not used for anything except cross-references between facts, for instance

fact(2, Allan, that(1))

could be used to indicate that the system knows that Allan knows that Janet knows it is raining. There must, clearly, be a way to represent facts which are available only to the system. The simplest way to do this is to permit an identifier such as "I" to stand for the system. It does not seem as though there is any need for specific rules about "I". The example below does not refer to "I". It is also necessary to allow facts to be known to more than one person. The simple indexing scheme used here makes it possible to refer to facts and rules that are known by everyone. For instance,

fact(3, @ALL, that(lives_with(@X, @Y)) → that(lives_with(@Y, @X)))

says that everyone knows that if X lives with Y then Y lives with X. As the system stands, facts which are known to a specific finite group of people can be recorded by enumerating them, as in

fact(3a, Allan, that(raining))
fact(3b, Julian, that(raining))
fact(3c, Ros, that(raining))
...

Rules which are true for some generic group of people, such as "all logicians know that automatic theorem proving for modal logic is hard", cannot easily be stated with the formalism as it stands. This is an area which needs more work.

(iii) Contingent facts are divided into 'identifying' and 'non-identifying' facts. Identifying facts are facts which state that someone not only knows that something is true, they also know how to refer to the things it is true of. Such facts are indicated by the use of the quantifier **what**, as in:

fact(4, Janet, what(sk1, phone_number(Alex, sk1)))

The notation is not perhaps as neat as it might be, but the general idea should be clear. The identifiers **sk1**, **sk2**, ... are used as Skolem constants to indicate existential quantification, the quantifier what is used to show that the person concerned has a rigid designator for the the thing referred to. Non-identifying facts are marked by the prefix **that**, so that

fact(5, Allan, that(lives_at(Alex, sk2)))

would indicate that Allan knew Alex lived somewhere, but that he did not know where.

Finally we come to the inference rules themselves. They are each prefaced by an English gloss. This gloss frequently starts with a parenthesis referring to a list of people. This is the list referred to in note (i) above with respect to whom the given rule is taken to operate.

Rule 1: (The people listed in ALL_KNOW know that) X knows P if P is a fact for X and they all know it is:

knows(@X, @P, @ALL_KNOW)
if
fact(@N, @X, @P),
check(@N, @ALL_KNOW).

This is used with facts like 1 and 2 above to deal with questions like "Does Allan know that Janet knows it's raining?". **check** is a predicate, defined below, which finds out whether people have direct access to particular facts.

Rule 2: (The people listed in ALL_KNOW know that) X knows P and Q if they know that X knows P and they know that X knows Q:

knows(@X, and(@P, @Q), @ALL_KNOW)
if
knows(@X, @P, @ALL_KNOW),
knows(@X, @Q, @ALL_KNOW).

This one simply reflects the assumption that everyone knows that if two things are both true then their conjunction is true. Since everyone knows it, it can be used at any point when we want to mimic someone else's reasoning.

Rule 3: (The people listed in ALL_KNOW know that) X knows Q if they know that X has access to a rule of the form "P implies Q" and they also know that X knows P:

knows(@X, @Q, @ALL_KNOW)
if
fact(@N, @X, @P → @Q),
check(@N, @ALL_KNOW),
knows(@X, @P, @ALL_KNOW).

This reflects the assumption that everyone knows modus ponens, i.e. that if you know P → Q and you know P then you can infer Q. It is important to note that although the rule allows the system to try to infer the antecedent using whatever means are available to it, the implication itself must be a fIfactfR. This is one of the places where we have chosen not to give the system the strongest possible inference rule - the given rule does support a lot of the ways that people use implications (for instance, that from P, P → Q and Q → R you can infer R), but it does not support arbitrary attempts to infer new contingent rules. This is a deliberate choice which could easily be altered. Allowing the system to try

application is to put a resource limitation on the inference process. This could easily be done, but it would add little to the interest of the example, and would just make things harder to follow):

know^OX, knows(@X, @F, @ALL_KNOW), @ALL_KNOW)
if
knowsKOX, @F, ©AIJLJKNOW).

This rule is allowed in by most people working in this area, to reflect the fact that you ought to be able to introspect on what you know. It is not used in any of the examples given below, but it is available for those few problems where it is actually useful.

Rule 5 (The people listed in ALL_KNOW know that) X knows that Y knows F if the list of people we get by adding X to ALL_KNOW know that Y knows F:

knowsCOX, knows(@Y, @F, @ALL_KNOW), @ALL_KNOW)
if
knowsCOY, @F, [@X I **@ALL_KNOW]).**

This rule allows us to delay worrying about how Y might infer F. All the other inference rules simply pass ALL_KNOW on unchanged until a specific contingent fact or rule is required. We thus do not need to check that everyone knows that Y can perform inferences. All that has to be checked is that he is known to have access to the specific contingencies.

Rule 6: (The people listed in ALLJKLNOW know that) X knows P, where P is a non-identifying proposition, if there is some other person, Y, who knows P, and the people in ALL_KNOW know that X knows Y knows it:

knows(@X, that(@P), @ALL_KNOW)
if
fact(@N, @Y, that(@P)),
check(@N, [@X I **@ALL_KNOW]).**

This comes from the fact discussed above that no-one can know anything which is untrue, so that if X knows that Y knows P then X must be able to infer P.

Rule 7: (The people listed in ALL__KNOW know that) X knows P, where P is a non-identifying proposition, if there is some other person, Y, who knows an identifying proposition about P, and the people in ALL_KNOW know that X knows that Y knows it:

knows(@X, that(@P), @ALL_KNOW)
if
fact(@N, @Y, what(@W, @P)),
check(@N, [@X I @ALL_KNOW]).

This is aimed at the case where Allan knows that Janet knows what Alex's phone number is. We cannot afford to derive the conclusion that Allan also knows what it is. but we do want to be able to infer that he at least knows Alex has a phone.

```
check(@N, [@C | @ALL_KNOW])
if
knows(@C, that(@N), @ALL_KNOW).
```

Contingent facts and rules - an example The first four facts refer to the specific knowledge possessed by Allan and Janet. They say that Allan knows that Alex and Trish live together; that Janet knows Alex's phone number; that Allan knows Janet knows Alex's phone number (but not that he knows it himself); and that Allan knows Janet's phone number:

```
fact(1, Allan, that(lives_with(Alex, Trish))).
fact(2, Janet, what(sk1, phone_number(Alex, sk1))).
fact(3, Allan, that(2)).
fact(4, Allan, what(sk2, phone_number(Janet, sk2))).
```

The next group of facts express various pieces of common knowledge. They can be used in chains of inference about the knowledge possessed by anyone at all, since the place for the name of the individual possessing the knowledge is filled by a variable (which will match the name of any specified individual):

Fact 5: (Everyone knows that) anyone who has a phone knows its number:

```
fact(5,@ALL,
       that(phone_number(@P,@N))
            →knows(@P,what(@N,phone_number(@P,@N))).
```

Fact 6: (Everyone knows that) if two people live together and one of them has a phone, the other one has a phone with the same number:

```
fact(6, @ALL,
     (that(lives_with(@X, @Y)) and that(phone_number(@X, @N)))
               → that(phone_number(@Y, @N))).
```

Fact 7: (Everyone knows that) if Y knows P then if X asks Y about P then X will know P as well:

```
fact(7, @ALL,
     knows(@Y, @P, @ALL_KNOW)
     → effect(ask(@X, @Y, @P), knows(@X, @P, @ALL_KNOW))).
```

It is assumed here that Y is perfectly co-operative.

Fact 8: (Everyone knows that) if X lives with Y then Y lives with X:

```
fact(8, @ALL,
     that(lives_with(@X, @Y)) → that(lives_with(@Y, @X))).
```

Note that this rule will lead to infinite regress unless the inference engine contains an explicit check.

Fact 9: (Everyone knows that) if X lives with Y then X knows X lives with Y:

```
fact(9, @ALL,
     that(lives_with(@X, @Y))
     → knows(@X, that(lives_with(@X, @Y)), []).
```

Fact 10: (Everyone knows that) if X can call Y then X can ask Y about anything he likes:

```
fact(10, @ALL,
     can(@X, call(@X, @Y)) → can(@X, ask(@X, @Y, @ABOUT))).
```

Fact 11: (Everyone knows that) if X knows Y 's phone number he can call Y:

```
fact(11, @ALL,
     knows(@X, what(@N, phone_number(@Y, @N)), []
          → can(@X, call(@X, @Y)))).
```

This rule set contains a number of bits of common knowledge, among which are a number of rules which give the preconditions and effects of the actions "calling" and "asking". These rules are clearly a long way from complete characterisations of these rather complex actions. It is also evident that while descriptions of actions in terms of effects and preconditions are useful to the standard AI planning algorithms, we have not given any implementation of a planner. In particular, the reasoning mechanisms outlined here are inadequate for investigating the effects sequences of more than one action. Moore (1984) does manage to mix chains of reasoning about other people's knowledge with investigations of the effects of sequences of actions, but, as noted above, his system does not seem to provide the performance that ours does. We would prefer to solve the problem of complex sequences of actions by adding to our system some mechanism for simulating the effects of the actions. Nonetheless, the ability of the system to answer the questions given below indicates that something like the current system might well by very useful for reasoning about how to plan to ask somebody something.

Examples of questions the system can answer As with the rule set, the examples are prefaced by English glosses. We do not pretend to have any programs which can translate in either direction between English and the notation used by the reasoning system. Readers who do not believe that there is any close correspondence between the formal notation and what we are offering as an English equivalent should disregard the glosses. We regret that we will probably not be able to convince such a reader that we have anything interesting to offer.

Question: whose phone number does Allan know?
Answer: Janet's.

```
GOAL(knows(Allan, what(@N, phone_number(@X, @N)), []).
```

ANSWER:

```
{knows Allan {what sk2 {phone_number Janet sk2}}} [].
```

Note that both the query and the answer have [] as their final component, indicating that the query was directly about what Allan knows, rather than about who else knows something about Allan's knowledge.

Question: what action does Allan know would lead to his knowing Alex's phone number?
Answer: he would know it if he asked Janet what it was.

```
GOAL(knows(Allan,
       effect(@ACTION(@X, @Y, @Z),
          knows(Allan, what(@N, (phone_number(Alex, @N)), □))), □))
```

```
ANSWER:
  {knows/4 Allan
          {effect {ask Allan Janet {what sk1 {phone_number Alex sk1}}}
                  {knows Allan {what sk1 {phone_number Alex sk1}} □}}
          □}
```

Question: what action does Allan know would lead to his knowing Trish's phone number?
Answer: he could ask Janet what Alex's phone number was. The chain of inference that leads to this conclusion depends on knowing that since Trish and Alex live together they must have the same phone number, so anyone who knows Alex's phone number will be able to provide the required information. Note that there is no need for Janet to know that Trish and Alex live together for this inference to work:

```
GOAL(knows(Allan,
          and (that(phone_number(Trish, @N)),
              effect(@ACTION(@X, @Y, @Z),
                    knows(Allan, what(@N, @E), □)), □))
ANSWER:
{knows/4 Allan
        {and {that {phone_number Trish sk1}}
            {effect {ask Allan Janet {what sk1 {phone_number Alex sk1}}}
                    {knows Allan
                           {what sk1 {phone_number Alex sk1}}
                           □}}} □}
```

This example should be taken as indication of the point at which the system starts to break down. The English gloss said "what action does Allan know would lead to his knowing Trish's phone number?". The query that was actually asked was more like "Does Allan know that Trish has a phone number and that there is some action which would enable him to find it out?" This is a consequence of our choice of inference rules. Stronger rules would have enabled the system to answer the right question. Note however that what Allan is to ask Janet is {what sk1 {phone_number Alex sk1}}, which can easily be glossed as "What's Alex's phone number?". This question correctly avoids mentioning Trish's number, since the available facts give no indication that Janet has ever heard of Trish, let alone that she knows she has the same phone number as Alex.

Question: what action does Allan know that he can perform which will lead to his knowing Trish's phone number?
Answer: he knows that he can phone Janet and ask her Alex's number. This is more complex than the previous case, since Allan has to prove not only that Janet will provide the information if asked, but also that he can ask her because he knows her number and hence can ring her:

```
GOAL(knows(Allan,
       and (and (that(phone_number(Trish, @N)),
                effect(@ACTION(@X, @Y, @Z),
                       knows(Allan,
                             what(@N, phone_number(Trish, @N), □)))),
            can(Allan, @ACTION(@X, @Y, @Z))), □))
```

ANSWER:

```
{knows/4 Allan
        {and {and {that {phone_number Trish sk1}}
                  {effect {ask Allan Janet
                                {what sk1 {phone_number Alex sk1}}}
                          {knows Allan
                                 {what sk1 {phone_number Alex sk1}}
                                 □}}}
             {can Allan
                  {ask Allan Janet {what sk1 {phone_number Alex sk1}}}}}
        □}
```

CONCLUSIONS

Previous workers in this area have concerned themselves largely with systems which will be able to draw all possible inferences about the states of knowledge of the various participants. Such systems have rather more power than the rule set given above, as is indicated by our last two examples, where the questions have had to be phrased extremely carefully. Nonetheless, our system has one great advantage: it is quick. Despite the fact that the inference engine which is used for finding and applying rules is, by the standards of most PROLOG compilers, rather slow (around 200LIPS on a VAX_11/750, compared to 5KLIPS upwards for commercially available compilers), the answers to the example queries took less than 10 seconds. Extending the set of inference rules would be expected to worsen the performance, but with finer tuning the background inference engine could be brought up nearer the 2KLIPS mark (it will always be slower than a pure PROLOG engine, since it needs to check for infinite loops). We cannot, in any case, ever afford to use a complete theorem prover in any autonomous or semi-autonomous system, since there will always be a risk of non-termination. It seems, therefore, entirely reasonable to aim for a system which will answer quite a lot of the questions we might want to ask, and doesn't take too long about it.

The main problems with theorem proving with modal logic arise from the presence of what we have referred to here as 'identifying facts'. If it were not for situations such as Allan knowing that Janet knows what Alex's phone number is without knowing it himself, the task would scarcely be more problematic than theorem proving for ordinary logic. The introduction of the quantifier **what** makes some of the problems more tractable. We have not, however, given much analysis of the *meaning* of this quantifier. It is easy enough to understand what it means to say "I know what Alex's phone number is" - it means that I know a sequence of numbers which when dialled will enable me to ring him up. The conditions under which I can truly say "I know who Alex's girlfriend is" are far less clear - does knowing her name suffice, or knowing what she looks like, or knowing that she is a lecturer at St Martin's School of Art? It is quite conceivable that I might know "who someone is" without ever having met them and without knowing their name, or what they look like, or indeed any true fact about them. The philosophical literature does not seem to have any very solid analysis of this problem. Most of the discussion has centred on the problem of "knowing who or what a name refers to", but an answer to this problem would probably provide an answer to our difficulty about "knowing who or what a definite reference refers to". Kripke's (1971) introduction of 'rigid designators' accounts for cases like phone numbers. As far as other entities are concerned, we leave it as an open problem. At any rate, even without a proper analysis of when we do and do not know what something is, we can be sure that our inference rule 7 (if X knows that Y knows **what(@W, @P)** then X knows **that(@P)**) is sound.

References

Appelt. D., 1985. Planning English sentences Cambridge University Press, Cambridge.

Konolige, K., 1982. A first-order formalisation of knowledge and action for a multi-agent planning system, in Machine Intelligence 10, (eds. Hayes, Michie and Pao), Ellis Horwood, Chichester.

Kripke, S.A., 1971. Semantical considerations on modal logic, in Reference and modality (ed Linsky). OUP, London.

Moore, R.C., 1980. Reasoning about knowledge and action SRI AI Center Report 191.

Moore. R.C., 1984. A formal theory of knowledge and action, in Formal theories of the commonsense world (eds. Hobbs and Moore), Ablex Pub.Corp., New Jersey.

Ramsay, A.M. and Barret. M.R., 1987. AI in practice: examples in POP-11 Ellis Horwood, Chichester.