KNOWLEDGE AND THE PROBLEM OF LOGICAL OMNISCIENCE

Rohit Parikh
Department of Computer Science, Brooklyn College, and
Mathematics Department, CUNY Graduate Center[1]

The notion of knowledge has recently acquired a great deal of importance in Computer Science, partly because of its importance in AI and expert systems, but also because of applications in distributed computing and possible, even likely relevance to cryptography. See [H2] for a representative collection of papers on knowledge.

An important outstanding problem in the logic of knowledge is the problem of logical omniscience, which plagues all the current logics of knowledge that are founded on the notion of a *possible world*. We assume in these logics the notion of a world, or situation which is possible relative to an individual (knower) i. The individual i knows a fact B iff B is true in all the worlds which are possible for i. It follows that if i knows A and also knows $A \rightarrow B$, then both A and $A \rightarrow B$ are true at all worlds possible for i and hence B is true at all the worlds possible for i so that i knows B. In particular, i knows all B which are logically valid. This principle, the principle of logical omniscience, is usually expressed as the formula

$$K_i(A) \wedge K_i(A \rightarrow B) \rightarrow K_i(B)$$

Now my knowledge may itself differ from world to world, and to deal with this fact we need the notion of equivalence of worlds. Two worlds are *equivalent* for me if they create the same evidence for me, and a world w' is *possible* for me at the world w iff w and w' are equivalent for me. Since w is always possible at w, if I know B at w, then B is true at w. In fact we get in this way a logic of knowledge that is sometimes called $S5$.

This kind of model and logic comes in very handy in distributed computing where we consider a bunch of processes connected in a network. At any moment of time, events may take place at zero or more sites, and a global history H is just the complete record of all events that have taken place at all sites. However, the *evidence* of a process is just its *local history* $h = f(i, H)$ and it knows a certain property A of H iff A is true not only of H itself, but also true of all H' such that $f(i, H)$ equals h. See ([PR], [CM]).

Unfortunately, it may well happen that deciding whether i knows a formula A at a history H may be an NP-complete problem, [MT], or even undecidable. Moreover, humans in general do not know the logical consequences of all their knowledge, even when such consequences are not too difficult to compute. Thus, in general, the amount of *actual* knowledge of facts is much less than the $S5$ logic would suggest.

There are, in fact, some proposals in the literature, based on syntactic treatments of knowledge, or more exotically, on the notion of an impossible possible world. See [H3] pages 7-9 for a brief overview of such attempts. However, none of these solutions is very natural and to find a better, more realistic approach, it is worthwhile being quite explicit about the problems that we want to solve. We propose a set of examples that any adequate treatment of knowledge must deal with and then propose one that, in our view, does this.

1) I point out to you that there is thick smoke coming out of a certain house and you conclude that the house is on fire. Here, if S stands for thick smoke and F stands

for fire, then you already knew $S \rightarrow F$, and on learning that S, you also know F. This situation, where knowledge is increased by the increase in evidence, is the only one that the $S5$ logic of knowledge can handle.

2) Nero Wolfe sits in his office and listens to a description of a scene of a murder, given by Archie Goodwin. Afterwards, Wolfe knows who the murderer is, but Archie does not. Wolfe is a fat and lazy man who almost never goes to the scene of crime, and has *no* evidence that Archie does not have. How can he know more than Archie?

3) I know the axioms of Peano arithmetic and also know that if Fermat's theorem is false, then this fact is deducible from the axioms. Hence if F is false, then I know $\neg F$. If, on the other hand, F is true, clearly I cannot know $\neg F$, and hence, by the $S5$ axioms, I know that I do not know $\neg F$. Combining this fact with my previous reasoning, I know F. Hence either I know F or I know $\neg F$. In other words, I know whether F is true. But of course, in fact I do not.

4) I ask someone who knows a little arithmetic but does not know any logic whether it is true that if it is the case that if $3 > 2$ then $3 > 4$, then it is also the case that $3 > 5$. She does not know and suspects me of trying to confuse her. After she learns the propositional calculus and the formalisation of the sentence above as $((P \rightarrow Q) \rightarrow R)$ where P is true and Q and R are false, she sees that the sentence is in fact true.

5) Here is a dialogue between a questioner q and a respondent r:

```
q: Do you  know the factorisation  of 143?
r:  Not off hand.
q: Is 11 a prime?
r: (After thinking  a little) Yes.
q:  Is 13 a prime?
r: Yes,
q: How much is 11 times 13?
r: Let us see; 11 times 10 is  110. 11 times 3 is 33. 110
plus 33 is 143.  Oh, I see.
q: Can you factorise 143 now?
r: Of course, it is 11 times 13.
```

There is nothing mysterious about this dialogue. However, note that q only asked questions. How then could he increase r's knowledge?

6) Do you know the last letter of the string *abracadabra*? Do you know if the number of letters in that string is prime? Notice that the first question was answered immediately, but the second one took a bit of counting. Nonetheless, you did not feel like answering either question with "I don't know", even though that answer was, strictly speaking, correct.

7) Some children having played in mud, have got their foreheads dirty. In fact, k of them have dirty foreheads. Every child can see everyone else's forehead, but not his/her own. The mother arrives on the scene and says "one of you has a dirty forehead." She then asks all the children one by one, if it knows its forehead is dirty. Strange to say, they *all* say, "I don't know."

In the conventional version of this puzzle, (see [HM]) all the children are supposed to be perfect ($S5$) reasoners, and it can be shown that the kth dirty child does know that it has a dirty forehead, but with real children, this is only likely if k is one. If k is two, the child who deduces that his forehead is dirty would be described as clever, and

Of course, everyone knows that real children will behave differently from idealised children. The problem is to find a theory of real children. To understand the puzzle properly, we should examine the conventional argument that the last dirty child uses to realise that its forehead is dirty. Let k be the number of dirty children. If $k = 1$, then the dirty child sees no one else who is dirty, and realises that it is itself the dirty child. Suppose now that we have justified the case $k = m$ and are considering the case $k = m + 1$. The $(m + 1)$th dirty child sees m dirty faces and knows that $k = m$ if it is itself clean and $k = m + 1$, if it is dirty.

It then reasons: if $k = m$, then the child before me would be the last dirty child and would have realised, using the argument for $k = m$, that its forehead was dirty. However, it said, "I don't know." It follows that $k = m + 1$ and *my* forehead must be dirty.

What happens to this argument in a realistic setting?

Suppose that I am one of the children; all the children that I can see, who have muddy foreheads, have been asked if their foreheads are muddy and have already said "I don't know." I know that if my forehead was clean, then the dirty child before me should have said, "my forehead is dirty." Since that child said "I don't know," then *if* that child was capable of making that inference and did not, then it must be that my forehead is dirty.

However, what if my forehead is in fact clean, and the failure of the child just before me to realise that his forehead is dirty, is merely due to a weak reasoning capacity? In that case I should not assert that my forehead is dirty. Thus my being able to make the right inference about my forehead depends on my trusting the reasoning ability of the child before me, and his depends in turn on his trusting the children before him. Also, perhaps he said "I don't know if my forehead is dirty," not because he lacked the capacity to make the right inference, but doubted the capacity of the child before him.

On the other hand, if the child before me *falsely* believes that all the children before him are perfect reasoners, then I am justified in trusting him *even* if his assertion is due to a false belief. Because I know that he will say "I don't know" iff he sees that my forehead is dirty. If he falsely believes that the children before him are perfect reasoners, then he runs the risk of saying "my forehead is dirty" when in fact it is clean. However, I run no risk.

I claim now that what is crucial here is the actual *behaviour* of the children, and not really their knowledge or even beliefs. Suppose, for example, that I instruct the children: "I am going to put mud on some of your foreheads, and plain water on others. I will put mud on at least one forehead. Then I will point at each of you, one by one. If you do not see any child with mud on his forehead that I have not pointed to, and no one has yet raised his hand, then you must raise your hand."

This procedure can be learned by the children much more easily than learning the $S5$ logic of knowledge. Moreover, if the children follow the instructions properly, then it will *always* be the case that the child who raises its hand will be the last muddy child. The earlier proof that the last dirty child knows that its forehead is dirty can be readily converted into a proof that the child that raises its hand will always be the last dirty child.

Does the child who raises its hand *know* that its forehead is dirty? Not necessarily: it might just think that we are playing some game. What we do have here is the correctness of a concurrent algorithm which depends on external knowledge (or $S5$ knowledge if you

knowledge. (See [PR] for a theorem connecting external knowledge with the possibility of safe concurrent procedures.)

Definition: i has *external* knowledge of A if A is true in all situations which are compatible with i's evidence.

Two other notions of knowledge that we can now define are l-knowledge, where l stands for "linguistic" and b-knowledge, where b stands for "behavioral".

i) i l-knows A if i has said "yes" to the question whether A is true and moreover, in all possible situations, i says "yes" to A only if i has external knowledge of A, and says "no" to A only when i has external knowledge that $\neg A$ holds.

ii) i b-knows A if there are three mutually imcompatible actions α, β, γ such that i does α only if A is true, and does β only if A is false, and moreover, i has just done α. (γ has no restrictions.)

Note that l-knowledge is knowledge of the *sentence* whereas b-knowledge is knowledge of the *proposition*. Thus if A and B are equivalent, then b-knowledge of A implies that of B. Not so for l-knowledge.

Thus a pigeon who has been trained to peck at a door with a square painted on it (rather than a circle, say), and has just pecked at a door with a square on it, can be said to b-know that the symbol is a square, though it can hardly respond to the question "is that a square?" Similarly, a thermostat that shows a reading of 70 degrees, b-knows the temperature, if it does not show this reading under other circumstances.

It is easily seen that l-knowledge implies b-knowledge. Just take $\alpha(\beta)$ to be the action of saying "A is true (false)", and γ to be the action of saying "I don't know". However, b-knowledge is more general in that it can be applied also to pigeons and to inanimate objects like thermostats and Turing machines. Thus suppose a Turing machine is programmed to recognise the language L. I.e., when started on a string x as input, it halts and prints a * on the tape only if x is in L. Then for a particular x in L, it has external knowledge that x is in L already when it starts, but it has b-knowledge that x is in L, only when it prints a * on the tape.

In general, b-knowledge should imply external knowledge, for it is presumably impossible to persistently do α only when A is true unless one has some sort of evidence that reveals the presence of A.

We now consider the question of *computing* knowledge, a question which involves resource bound considerations. In general, the goal is to compute external knowledge, but it may be incomputable or intractable and l or b-knowledge of facts that one may succeed in computing will generally be less than external knowledge. But knowledge of other people's knowledge that one may have, could *exceed* what is possible with external knowledge; for example, it is possible for me to have l-knowledge that you do not know Fermat's theorem, but if the theorem is true, then it is impossible for me to have external knowledge that you do not have external knowledge of it.

The dirty children puzzle shows, moreover, that a concurrent algorithm which works if everyone acted according to external knowledge, might not work if l-knowledge is substituted for it.

Let us return to a discussion of the role of resource bounds which, we claim, are implicit in every query about knowledge.

Suppose I ask someone, "do you know what time it is?" He is likely to look at his watch and say "four PM", or whatever his watch shows. At this point it would be silly

why didn't you say so?" Presumably he took my inquiry, not really to be an inquiry about his state of knowledge, but about the time. What happens is that he is carrying out an algorithm which terminates with his knowing the value of the time. If I now ask the same question of a person not wearing a watch, she is likely to say, "I don't know". If, however, I say, "Do you know what time it is? I have a plane to catch at six". Then she may suggest calling the special number for time, or remember someone around who is wearing a watch. Thus the algorithm is carried further if the likely benefit from the answer exceeds the cost of carrying out the algorithm. Note also, that unlike a computer, when a person is asked the same question twice, and the question was answered successfully the first time, then the second query is answered quite fast. This clearly points to some database that has been updated in the meanwhile.

Definition: A *knowledge algorithm* consists of a database together with a procedure that takes as input a question (say the truth value of some formula) and some resource bound, and operates on the question and the database upto some point determined by the value of the resource bound. Then either an answer is obtained or the bound is exceeded. In the first case, the answer is given and the database may be updated, even if the answer depended logically on evidence already at hand. In the second case the answer "I don't know" is given[2]. The database may also be updated as a result of information received from the outside.

We illustrate this by describing a possible knowledge algorithm for factorisation. The database in this case consists of a finite sequence of primes, not necessarily in increasing order. A query consists of a pair n, b where n is the number to be factorised and b is a resource bound. The algorithm uses some test for primality, perhaps a probabilistic one. On receiving the query, the number is first tested for divisibility by a number in the database. If the answer is positive, say n is divisible by p, then p is set aside as a factor and n is replaced by n/p. If no prime in the database divides n, then n is tested for primality. If n turns out to be a prime, then the factorisation of n is trivial and n *is added to the database.* If n is not a prime and is not divisible by any prime in the database, then some dreary routine for factorisation is started. If, at any time during this process, the bound b is exceeded then the answer "I don't know" is given.

We now look at the various problems that we have discussed earlier. Problem l is also solved by the usual $S5$ logic, but the current framework allows us to see that you *might not* see that there is a fire. Consider number 2. The explanation is that Nero Wolfe uses a more efficient algorithm for answering queries. The possibility that he knows more *initial* facts is discounted by the fact that his explanations can be followed by Archie. Problem 3 is now quite straightforward. Perhaps my algorithm for answering queries about number theoretic questions is merely to enumerate all possible proofs in Peano arithmetic. In that case, if F is false and there is a small (relative to my resource bound) counterexample, then I will find it. However, we know in fact that there is no small counterexample, and thus my ignorance of the truth value of F is easily understood.

Consider problem 4. In this case the person who knows arithmetic has an algorithm for answering inequalities, and perhaps simple truth functional combinations. However, the question asked falls outside the domain of the algorithm. When she learns the propositional calculus, she changes her algorithm and can now answer the question asked.

Problem 5 illustrates the use of a database. At first the person asked only has the

[2]This answer really should be distinguished from the same answer given when one knows that one

possibility of trying all possible factors up to the square root of 143 and does not want to bother. (If money were offered, then this would increase the value of an answer and increase the value of b.) However, the rest of the dialogue puts 11 and 13 in the database, and the same question is now easily answered. Similarly, in public key cryptosystems, the private key is part of the database so that two people with different databases will be in different positions regarding the decoding of some message. It is also clear that if one party knows a fast algorithm for factorisation and the other party does not even know that there is one, then their situation is quite unsymmetric, a fact that the $S5$ logic cannot represent.

Problem 6 illustrates the fact that any query is accompanied by a computation. In those cases where one already "knows" the answer, i.e. the answer is stored in a database, the query can be answered in logarithmic time (in the size of the database), and this will usually be swamped by the linear time (in the size of the query) needed to hear the question. For giving the last letter of the string *abracadabra*, the computation is fast and one notices no hesitation. For deciding whether the number of letters is a prime, more time is needed, and one is conscious of the computation. A similar situation arises with understanding English. In most cases, the parsing can be done in real time, but a complicated sentence may cause one to think a bit.

We finally come to the dirty children puzzle. To understand this puzzle properly, we have to distinguish two different kinds of "I don't know" answers. If I ask you of some large odd number if it is a prime, you will say "I don't know", meaning really that you could find out with enough time, but you don't know the answer offhand and do not want to bother to compute. Suppose, however, that I ask you if Rajiv Gandhi's age is a prime number, your answer "I don't know" probably means that you lack external knowledge of his age. Hence you also lack l-knowledge of the answer, and your ignorance is stable relative to further computation.

Under special circumstances, external knowledge or the lack of it may be decidable relatively cheaply. If a person knows this, then he might respond to a query by carrying out the computation till it terminates in a "yes", "no", or a stable "I don't know". In the dirty children puzzle, there is such an algorithm which works, provided that all the children carry it out faithfully. This is why we have a proof that the kth child "knows" that it is dirty, provided that all children are perfect reasoners. If they are not, then the kth child does not even have external knowledge that he is dirty, because part of his evidence is the answers given by the other children, and there is no way to interpret this if their l-knowledge is properly less than their external knowledge.

We have so far talked only about how inquiries about knowledge may be answered. But what then is *knowledge*? Given that a person has a knowledge algorithm, we may perhaps like to say that A is *knowable* to him if the algorithm with input A will terminate for some large b and give a "yes" answer. The set of all knowable A need not follow any pattern in general, but we can show the following theorem:

Theorem: Given a knowledge algorithm α, there is an algorithm β, extending it, such that the set of β-knowable A obeys the $S4$ logic.

The theorem follows from the fact that the set of knowable A under α is r.e. and hence the closure of it under the $S4$ axioms and rules (in the propositional or first order versions) is also r.e. The theorem will not necessarily hold if $S5$ closure is wanted because of the $S5$ requirement that one knows which facts one does not know.

Conclusion: We have tried in this paper to make a case that real knowledge is not a *set* but a *behaviour* ([R] pp. 26-32 makes a similar point) and that positive answers to questions like "is A true?" are given for A lying between the already known set (A in the database) and the set of externally true statements. Moereover, in response to a particular query even for an *externally knowable* A not in the database, the answer may be "I don't know" or "Yes", depending on the value of the resource bound and whether the algorithm followed is complete relative to the external knowledge. The definitions of knowledge we have proposed may not strike the reader as all that surprising. However, they seem to be new, they handle the verious problems satisfactorily and they are relatively simple.

What about notions like polynomial time knowledge? We feel that "polynomial time" is a theoretical notion whose primary interest is due to the fact that it leads to a nice theory and important mathematical questions. In practice our knowledge does not follow it very much even in areas that have already been thoroughly investigated. For example, matrix multiplication is in polytime while propositional tautologies are not, (so far) and first order logic is even undecidable. Nonetheless most of us know far more valid formulae of the propositional and first order logics, than know products of matrices, nor would we be enthusiastic about multiplying two 100×100 matrices by hand.

Moreover, given any particular sentence A, it belongs to many polytime sets as well as to many sets of much higher complexity classes. Thus complexity of classes can't always tell us much that is meaningful about individual questions. We may like to study complexity theory, but we should not lose sight of the fact that the paradigm situation is where we have a particular question and a particular (perhaps rough) resource bound, and want to know if this question can be answered in this context.

We thank Paul Krasucki for comments and Anil Khullar for help with LaTeX.

References

[CM] M. Chandy and J. Misra, "How Processes Learn", *Proceedings of the 4th PODC* (l985), 204-214.

[H] J. Hintikka, *Knowledge and Belief*, Cornell U. Press, l962.

[H2] *Reasoning about Knowledge*, Ed. J. Halpern, Morgan Kaufman, l986.

[H3] J. Halpern, "Reasoning about Knowledge: an Overview", in [H2] above, l-l7.

[HM] J. Halpern and Y. Moses, "Knowledge and Common Knowledge in a distributed Environment", *ACM-PODC l984*, pp. 50-6l.

[MT] Y. Moses and M. Tuttle, "Programming simultaneous actions using common knowledge" *27th IEEE-FOCS* (l986) pp. 208-221.

[Pa] R. Parikh, "Logics of Knowledge, Games and Dynamic Logic", in *FST/TCS 4*, Springer LNCS no. 181, pp. 202-222.

[PR] R. Parikh and R. Ramanujam, "Distributed Processing and the Logic of Knowledge", in *Logics of Programs '85* Springer LNCS no.l93, pp. 256-268.

[R] G. Ryle, *The Concept of Mind*, Barnes and Noble, originally published in 1949 by the Hutchinson company.