Reasoning About Deduction With Unknown Constants

Andrew Haas

Department of Computer Science, University of Rochester

fbs1;ract. An intelligent agent must plan future deductions and anticipate what other agents will deduce from their beliefs- Creary (1979) proposed to do this by simulation. Often the agent's future beliefs, or the beliefs of other agents, involve terms unknown to the agent at simulation time. This paper shows how to extend Creary[1]e technique to handle this case.

Suppose Mary knows Jane's phone number, and knows also that Jane's number is the same as John's. How does one show that Mary can infer what John's number is? Consider another problem: a robot has a piece of paper with John's number written on it, and wants to know John's number. By looking at the paper it can learn what's written there. How will it infer that this number is John's phone number? Both of these problems involve reasoning about a deduction whose premisses contain an unknown constant. In the first problem the unknown constant is Jane's phone number. Mary knows it, but the agent who is reasoning about Mary doesn't. This agent would know the number only if he believed a sentence of the form "Mary knows that Jane's number is n", where n is the constant that Mary uses to name Jane's number. In the second problem the unknown constant is the number written on the paper. The robot expects to know it after looking at the paper, but doesn't know it at planning, time.

Moore (1980) has devised a method of reasoning about deduction that handles unknown constants. But his method is too strong. It allows one to show that an agent knows any sentence that can be proved from his knowledge, no matter how difficult the proof is. Creary (1979) proposes a method of reasoning about deduction that avoids this problem. But his method can't handle unknown constants. This paper describe the difficulties that arise in applying Moore's theory to the problem of the robot finding John's phone number. It presents a version of Creary*s method. Finally it shows how to extend Creary's method to handle premisses with unknown constants. The result is a scheme that achieves the power of Moore's method and avoids its problems.

Consider the problem of finding out what John'8 phone number is. The robot knows that whatever is written on the paper is John's number. By looking at the paper it finds out that some number n is written there. These two facts imply that John's number is n. Since the robot always knows all logical consequences of its knowledge, it knows that John's number is n as soon as it learns that n is written on the paper. So in Moore's system the robot's plan for learning John's number consists of the single step of looking at the paper.

Imagine a real robot trying to execute this plan. It sends a command to its TV camera to point at the paper. In the input buffer appears a data structure that says that n is written on the paper. And here the plan stops. The robot still has to perform a short computation, inferring "John's number is n" from "n is written on the paper" and "Whatever is written on the paper is John's number". But Moore's plan omits this step, because the agents in his theory make all possible inferences from their knowledge automatically. If a robot can't do this, it should not use Moore's theory to plan its own inferences.

An alternative to Moore's theory of knowledge and belief is the syntactic theory. It says that a belief or piece of knowledge is a sentence (Moore and Hendrix, 1979)- Suppose these sentences are in first-order logic (motivation for this appears later). If a robot plans to acquire certain beliefs, it is thinking about its own beliefs. So the language in which those beliefs are expressed must be capable of talking about itself. In particular, it must contain names for its own expressions.

To each constant of the language assign a name, formed by appending a subscript 1 to that constant. Thus if "John" is a constant and denotes a man,

"$John_1$" is a constant and denotes "John". Now consider the symbols that are used to build new expressions - the predicate letters, function letters, connectives and quantifiers. Call these symbols constructors. Thus the sentence "or(p, q)" contains the constructor "or", and its arguments are "p" and "q". To each n-adic constructor assign an n-adic function letter, again formed by appending a subscript 1. Suppose "z" is an n-adic constructor. Then the function letter "z," denotes a function that maps n expressions e1...en to the expression whose constructor is "z" and whose arguments are e1...en. Thus if "and" is a connective, "and_1" denotes a function that maps the sentences "p" and "q" to the sentence "and(p, q)[tf].

So the representation of "The robot believes that John's number is 444-1212" is

(1) believe(robot,
$equal_1$ (phone-j ($John_1$),
$444\text{-}1212_1$))

The second argument of the function letter "believe" denotes the sentence

(2) equaKphone(John),444-1212)

Creary suggests essentially this method of quotation. He goes on to suggest simulation as a method of reasoning about deduction. It works like this. Begin with a set of sentences of the form

believe(A, x)

where x is quoted. By stripping off the subscripts you can reconstruct the sentences that A believes. Then collect these in a separate data base. Try to prove the desired theorem in this data base. If you succeed, infer that A can prove this theorem if he wants. Use some measure of the effort expended to predict how long A will take to get this result. This method of reasoning about deduction does not allow us to prove that agents know all consequences of their knowledge.

This won't work if A's beliefs contain constants unknown to the simulator. Without knowing these constants the simulator can't build a data base containing A's beliefs. One can solve this problem by taking advantage of the following property of first-order logic (proved in (Tennant 1978)):

If any substitution of terms without variables for constants is applied to the conclusion and premisses of a first-order proof, the result is again a valid first-order proof. (There are exceptions, such as the rule of all introduction in natural deduction. My technique can be extended to handle these cases too.)

This implies that one can use dummy constants to represent unknown terms in another agent's beliefs without disturbing the process of simulation. Whatever proof you find when you simulate will work just as well when the dummies are replaced by the real terms.

Consider the problem of finding John's phone number. The robot expects that after looking at the paper it will believe that n is written on the paper, where n is the arabic numeral for John's phone number. To represent this expectation one must be able to describe this n without knowing which numeral it is. Define the predicate "arabic" so that "arabic(x, y)" is true if x is an integer and y is the arabic numeral for x. Arabic numerals are constants of the language. The following sentence says that the robot believes that n is written on the paper, where n is the arabic numeral for John's phone number:

(3) exists(n,
and(believe(robot,$written_1$(n)),
arabic(phone(John), n)))

The robot believes that whatever is written on the paper is John's number, that is

(4) believe(robot, all.Cv*,
$implie8_1$ (writte^ (v^),
equal.(phone.(John-), v.)))

Use the dummy constant N to represent the unknown arabic numeral. Then the data base for simulation contains the following sentences:

(5) written(N)
(6) all(v, implie8(written(v),
equal(phone(John), v)))

Applying the rules of all elimination and detachment gives

(7) equal(phone(John), N)

Substitute the arabic numeral for John's phone number for the dummy constant N in the premisses and conclusion of this proof. The result is a valid proof. The premisses of this new proof are the beliefs described in (3) and (4). Its conclusion is described by

```
(8) exists(n,
    and(believe( robot,
                 equal,(phone^(John^),  n)),
        arabic(phone(John),  n)))
```

This is the belief the robot wants. This argument shows that the robot can obtain this belief by applying all elimination and detachment to the beliefs described in (3) and (4). When the robot sees the number written on the paper it will use this proof to infer that that is John's phone number- The example of Mary deducing what John's phone number is can be done the same way. In this case the inference rule is substitution of equals rather than detachment.

In conclusion, it is possible to use simulation for reasoning about a deduction whose premisses contain constants unknown to the simulator. This technique combines the advantages of Creary's technique (one can judge how difficult an inference is) and of Moore's technique (unknown constants cause no problem). A program can use the technique either to plan its own future deductions or to anticipate what other agents will deduce from their beliefs.

References.

Creary, Lewis. "Propositional Attitudes: Pregean Representation and Simulative Reasoning." Proc. IJCAI 79, p. 176.

Moore, Robert. "Reasoning About Knowledge and Action." Ph.D. thesis, M.I.T. 1980.

Moore, Robert and Hendrix, Gary. "Computational Models of Belief and Seamntics of Belief-Sentences." SRI Technical Note 187, 1979.

Tennant, Neil. Natural Logic. Edinburgh, 1978.