Communication and Inference through Situations

Hideyuki Nakashima
Electrotechnical Laboratory
1-1-4 Umezono
Tsukuba Ibaraki 305
Japan
nakashim@etl.go.jp

Stanley Peters
CSLI
Ventura Hall
Stanford, CA 94305
USA
peters@csli.stanford.edu

Hinrich Schiitze
CSLI
Ventura Hall
Stanford, CA 94305
USA
schuet ze@csli. stanford.edu

Abstract

In this paper, we show how to use common knowledge computationally in solving problems involving cooperation of multiple agents, when common knowledge is available. We will explain why a procedural approach to common knowledge is better suited to solving multiple-agent problems than a static one.

We show, even if one can never prove that common knowledge has been attained ([Halpern and Moses, 1990]), that assuming it has been attained is often safe and efficacious. The ability to detect fairly reliably when certain conditions are not met suffices as a guideline for when to assume something is common knowledge. In principle, the problem of when one has individual knowledge is about as difficult.

We use the *situation oriented* programming language PROSIT. By combining reasoning *about* situations and *in* situations, PROSIT makes possible an especially intuitive and simple solution of hypothetical reasoning problems involving common knowledge ([Nakashima and Tutiya, 1991]).

1 Introduction

Treatment of common knowledge is important in connection with multi-agent systems. When several or many agents cooperate to solve a problem that cannot be solved by any single one, it is essential for them to have models of other agents and to communicate with one another. In treating such systems, we need abstractions of a higher level than individual knowledge/beliefs, namely the knowledge/belief of groups of agents. Indeed, [Levesque *et* al., 1986] claims that the concept of joint intention is necessary to formalize problem solving by a group.

In this paper, we will show that

- common knowledge can be used algorithmically in solving problems cooperatively, and
- common knowledge is assumed rather than established.

To represent common knowledge, we use the framework of situation theory, which has been developed to provide a powerful logical foundation for analyzing information flow when partiality of information is crucial ([Barwise, 1989], [Devlin, 1991]), as it is in distinguishing between individual and common knowledge.

We will compare our model of common knowledge with Barwise's ([Barwise, 1989a]) and show why his model is not the best to make use of common knowledge in inference. We then apply our model to the three wisemen problem.

1.1 Situation Theory

Fundamental notions of situation theory include item of information (called an *infon)* and *situation* (a part of the world capable of making an infon a fact). A typical infon would be that a particular relation holds (alternatively, does not hold) of particular objects. For instance, the information that Pat, a card player, has the ace of clubs is the infon

```
(has Pat a_clubs).
```

An infon is a fact, if it is, by virtue of some part of the world making it so, i.e. some situation *supporting* (often written f=, but we write !=) the infon. For instance, a situation g in a card game in which Pat has the ace of clubs supports this infon, i.e.

```
(!= g (has Pat a.clubs)).
```

The knowledge that this is so will likely be included in Pat's individual knowledge - information which we can write as the two infons

```
(knows Pat pk)   (!= pk (has Pat a_clubs))
```

where pk is Pat's individual knowledge situation. If the information is possessed not only by Pat but is moreover common knowledge of Pat and Max, the other player in the game, we have the infons

```
(knows Pat ck)   (knows Max ck)
(!= ck (has Pat a_clubs))
```

for the common knowledge situation ck, which is a subsituation of both pk and Max's individual knowledge situation ink. That ck is common knowledge is itself common knowledge, i.e.

```
(!=  ck (knows Pat ck))
(!=  ck (knows Max ck)).
```

The last three of these propositions form the heart of Barwise's logical analysis of common knowledge [Barwise, 1989a].

Note crucially that different situations may be responsible for the facthood of different items of information, i.e. different situations may support different infons that are factual. The fact that Max actually has the ace of spades, i.e. `(!= g (has Max a_spades))`, might not be known to Pat

```
(no (!= pk (has Max a_spades)))
```

and therefore may not be common knowledge of Max and Pat.

```
(no (!= ck (has Max a_spades)))
```

But it can still be individual knowledge of Max's.

```
(!= mk (has Max a_spades))
```

Situations are often closed under constraints. For instance, the card game situation g respects the constraint that each card is held by at most one player; thus Pat having the three of clubs involves Max not having that card.

```
(<= (no (has Max 3_clubs)) (has Pat 3clubs))
```

Since g respects this constraint, if g is consistent and

```
(1) (!= g (has Max 3_clubs))
```

it is valid to infer

```
(2) (no (!= g (has Pat 3_clubs)).
```

If Max knows he has the three of clubs, i.e.

```
(3) (!= mk (has Max 3_clubs))
```

he will be aware of (1). If he also knows that g is a consistent situation respecting the constraint under consideration, Max can infer (2). From this he can go on to conclude

```
(!= g (no (has Pat 3_clubs)))
```

using the definiteness of g with regard to who holds which cards. In consequence of the second fact above it can thus come to be that

```
(!= mk (no (has Pat 3_clubs))).
```

We will turn soon to a specific problem mixing individual and common knowledge of agents in a more challenging way, showing how to formalize the problem in terms of situation theory and how this helps in solving it computationally. But let us first introduce the programming language we use for implementing our proposal.

2 PROSIT

In this section we give a short introduction to PROSIT, the programming language we use, and the reasons why it is especially suitable for human-like reasoning.

PROSIT (PROgramming in Situation Theory) is a programming language implemented in CommonLisp that has many features of Situation Theory built into it ([Nakashima *et* al., 1988]). In particular, a program can build up and navigate within situation structures; infons can be asserted in situations; and constraints hold between different situations in the situation structure. The inference engine is similar to a Prolog interpreter.

The numbers in the following description of the main constructs of PROSIT refer to Figure 3.

- Variables are marked with a *.
- (1): `<=` is a constraint (implemented as a backward chaining rule): `(white *x)` holds if the infons in (2) - (12) are true.
- (2): `(me *x)` binds `*x` to the current situation (reflection).
- (8): `!` asserts its argument into the current situation.
- (8): `[_` is the subchunk relation. The first argument is part of the second (both arguments are situations). Example: After the assertion (8), whenever `(rel a b)` holds in `c-channel`, `(!= c-channel (rel a b))` holds in `*y`. No inheritance relation holds between the two situations.
- (9): `@<` is the inheritance relation (or subsituation relation). Any infon holding in the first situation will also hold in the second.

The following properties make PROSIT especially suitable for writing programs doing human-like reasoning:

Inconsistencies. Reasoning is different from applications like numerical or accounting programs in that inconsistencies can occur. Moreover, inconsistencies play an active role; they are used to get at new information.

PROSIT's situation structure facilitates using contradictions for this purpose: contradictions can be kept local, preventing them from generating unsound inferences; and they can be reflected upon in a situation that the contradictory one is a subchunk of.

Direct programming. PROSIT allows navigation in the situation tree, i.e. the situation tree is not only a data structure, but also a *programming structure.* One can, therefore, always program from the point of view of the person whose reasoning has to be modelled. Instead of programming from an abstract level ("A thinks that B believes that C knows ...") reasoning can be *directly* translated into PROSIT code. This results in easy-to-write and well-structured programs.

Situated programming. In PROSIT we can state constraints in a general form that will be used differently in different situations by picking up the particular information available in a situation. This "lean" programming is possible because situations contain only the information pertinent to them; computations can be made local and efficient, ignoring what is happening "further up" (cf. [Nakashima and Tutiya, 1991]).

An example of how a general constraint picks up information from a situation is the special predicate (me *x) that binds its variable to the current situation.

3 Two Computational Approaches

to Common Knowledge

In this section, we investigate two different approaches to handling common knowledge in computation: a static and a procedural one.

3.1 The Static Model

The static representation consists essentially of the three following lines ([Barwise, 1989a]):

```
(1) (!= ck σ)
(2) (!= ck (knows Pat ck))
(3) (!= ck (knows Max ck))
```

σ is the "nucleus" of the common knowledge, the infon that the persons Pat and Max both know. What makes this situation common knowledge, as opposed to merely shared knowledge, is the fact that Pat and Max know of each other's knowledge *and are aware of this.*

We can translate this straightforwardly into PROSIT as follows:

- We need not use the fact that Pat and Max know **pk** and **mk** respectively. We simply express that Pat knows the fact σ as `(!= pk σ)`. (Recall that **pk** and **mk** stand for the knowledge of Pat and Max.)
- With this simplification, (2) and (3) are equivalent to `(!= ck (!= pk ck))` and `(!= ck (!= mk ck))`, respectively. This is precisely the definition of the subchunk relation, which we therefore use for the representation of (2) and (3).
- We incorporate the fact that what is common knowledge is also individual knowledge of each member of the group by making **ck** a subsituation of **pk** and **mk**.

The program, then, is the one shown in Figure 1. Any of the queries in Figure 2 will be answered yes. Notice that Pat and Max know not only everything about each other's knowledge but are also well informed about what is common knowledge.

We now come to to the crucial test for the usefulness of this representation: the integration of private and common knowledge. We try to apply the static approach to the Conway paradox:

> During a card game both Max and Pat have an ace. If asked whether they have any knowledge about the other person's cards they will answer no and this answer won't be different if the question is repeated; but if someone tells them "At least one of you has an ace", a fact they can infer from their own cards, the answer will be "No" for the first answering (Max) and "Yes, he/she has an ace" for the second of the two (Pat).

Pat reasons as follows to the conclusion. If Max didn't have an ace, he could infer that I have one because he knows that there is at least one ace. But then he wouldn't have answered "No" when he was asked whether I had an ace. Hence, the assumption that he doesn't have an ace is wrong.

How can we do Pat's reasoning within the data structures sketched above? The argument hinges on the assumption (no (has Max ace)). Where do we put this assumption? We're confronted with the following dilemma: If we put it in ck or mk, then (no (has Max ace)) will become part of Max's knowledge, which is clearly inadequate. If we put it in pk, then (!= pk (no (has Max ace))) becomes part of Max's knowledge, since pk is a subchunk of ck and what holds in ck is inherited by mk. This is inadequate too: we don't want Pat's assumptions to influence Max's knowledge. If we put the assumption in a situation s outside of ck, our common knowledge representation cannot be used: If we want to do situated reasoning in s, the common knowledge facts are not available.

So we cannot prove the desired proposition using a proof by contradiction. We do not deny the obvious fact that, in principle, one can derive any valid proposition without resorting to a proof by contradiction. For every proof that makes an assumption that is then shown to lead to an inconsistency, there is a "positive" argument proving the same proposition.

To derive the fact that Max has an ace by using positive arguments exclusively, requires common knowledge to include all possible alternatives:

```
(or (and (has Max ace) (has Pat ace))
    (and (no (has Max ace)) (has Pat ace))
    (and (has Max ace) (no (has Pat ace))))
```

The reasoners have to have the necessary common knowledge "prior to" reasoning, instead of deriving it only when needed. This demand is too strong in general for intelligent agents.

We believe that proofs by contradiction model much human reasoning more closely than positive proofs. We will see that in the case of the wisemen problem the natural way to solve it is a proof by contradiction, not a positive argument.

3.2 The Procedural Model

Taking into account the lessons we have learned from the static approach we propose a model with the following three key features:

1. The "common knowledge" situation is flat and non-circular.
2. Whereas in the static approach there was just one situation for each person, in our model all instances of knowledge situations are different. E.g. Max's knowledge and beliefs and Pat's knowledge and beliefs of what Max knows will be realized by distinct situations.
3. The common knowledge feature in our model will be provided by explicitly asserting a link between each instance of a knowledge situation and the "common knowledge" situation so that its contents can flow into this instance.

Now the character of common knowledge is not represented by a static, circular "common knowledge" situation as in the static approach. The common knowledge situation here is a *communication channel* that contains all information that is known to be commonly accessible. The circularity of common knowledge is realized

```
(! ([_ ck ck))                          ; declare ck to be self-referential
(! (!= ck ([_ ck mk)))                  ; subchunk relations: the common knowledge is a
(! (!= ck ([_ ck pk)))                  ;  subchunk of both Max's and Pat's knowledge
(! (!= ck (@< ck mk)))                  ; subsituation relations: Max and Pat's knowledge
(! (!= ck (@< ck pk)))                  ;  inherit from the common knowledge situation
(! (!= ck (has Pat ace-of-clubs)))      ; let (has Pat ace-of-clubs) be our sigma
```

Figure 1: Implementation of the Static Model

```
(!= ck (!= pk (has Pat ace-of-clubs)))
(!= ck (!= mk (!= pk (!= mk (has Pat ace-of-clubs)))))
(!= ck (!= mk (!= ck (has Pat ace-of-clubs))))
(!= ck (!= pk (!= ck (!= ck (!= pk (has Pat ace-of-clubs))))))
```

Figure 2: Inferences in the Static Model

procedurally by explicitly giving each knowledge/belief situation access to the communication channel (3).

Feature (2) reflects our view that common knowledge is generally interesting only if combined with other non-common, i.e. personal knowledge. If we want to mix personal and common knowledge to actually make use of what is mutually believed, we have to provide a new situation for each additional piece of personal knowledge that is added. This prevents assumed or wrong information from getting where it might lead to unsound inferences.

In the next section we show how we use the procedural model to implement our solution to the wisemen problem.

4 The Three Wisemen Problem

> Three wisemen are sitting at a table, facing each other, each with a white hat on his head. Someone tells them that each of them has a white or red hat but that there is at least one white hat. Each wiseman can see the others' hats but not his own. If a fourth person asks them whether they know their own colour, then the first two wisemen will answer no, but, after that, the third one will answer yes.

Note the parallel with the Conway paradox:[1] The wisemen already know *individually* that there is at least-one white hat; again individual knowledge by itself would be useless, only the combination of private and common knowledge yields the desired result.

We analyze the problem as follows:

- common knowledge:
  1. There are three agents A, B and C; all are wise, i.e. good reasoners and thoughtful; each wears either a red or a white hat; at most two hats are red.
  2. Each can see the others' colours.
  3. None can see his own colour.
  4. Each can hear and understand all utterances.
  5. All these facts are common knowledge.
- individual knowledge:

  For each agent:

  A: B is white. C is white.

  B: A is white. C is white.

  C: A is white. B is white.

The problem has the following features:

1. All agents have the same inference mechanism.
2. All agents know (1).
3. Knowledge of the facts is somewhat different for each agent.
4. These facts cannot be directly transferred among agents.
5. Information is collected by observing the others' behaviour.

Let us now examine how to implement this analysis in PROSIT. For lack of space, we will only consider the two central constraints which are shown in Figure 3.

Note that we use situations to represent one person's internal model of the others. Situations here are not necessarily actual situations in the world. They are rather abstract ones (see [Nakashima and Tutiya, 1991] for a justification of using abstract situations in reasoning). In Figure 3, the situation bound to *y is such a model. It is a situation internal to *x and has no necessary connection with the actual *y (which is another wiseman).

*x knows that *y perceives the third wiseman *z. *x's mental model of *y therefore has to contain all facts about *z that can be inferred from (in this case) seeing *z. transfer-knowledgejBtbout-third makes sure that these perceivable facts are added to *x's mental model of *y. Since we don't want to get into the intricacies of perception, we have written (colour *colour) into the procedure, colour being the only relevant perceivable property in this case.

Other parts of the program will take care of the individual knowledge and the right content of the common knowledge situation. Note that the common knowledge

[1]We use the Conway paradox, which is a kind of "Two Wisemen Problem," in order to make the argument against the static model easier to understand.

```
(<= (white *x)                      ; ( 1) try to deduce (white *x)
    (me *x)                         ; ( 2) -|
    (wiseman *y)                    ; ( 3)  | bind *x to the wiseman doing this
    (wiseman *z)                    ; ( 4)  | reasoning and *y and *z to
    (not (= *x *y))                 ; ( 5)  | the two other wisemen
    (not (= *z *x))                 ; ( 6)  |
    (not (= *z *y))                 ; ( 7) -|
    (! ([_ c_channel *y))           ; ( 8) make channel accessible to hypothetical
    (! (@< c_channel *y))           ; ( 9)  reasoning situation for *y
    (! (!= *y (red *x)))            ; (10) assert assumption (red *x) in *y
    (transfer_knowl_on_3rd *y *z) ; (11) add in *y what *x and *y know about *z
    (inconsistent *y)               ; (12) if the assumption produces an inconsistency in *y,
    ))                              ; (13)  it is wrong, which proves its opposite: (white *x)

(<= (transfer_knowledge_about_third *y *z)
    (or
     (and
      (colour *colour)              ; choose a colour to bind *colour (example: red)
      (*colour *z)                  ; if e.g. (red *z) is true
      (! (!= *y (*colour *z)))      ; then assert that *y knows this fact
      )
     (true)                         ; don't do anything if there's nothing
     ))                             ;  to transfer
```

Figure 3: Implementation of the Procedural Model

situation is more like a communication channel here: it is non-circular. It contains all commonly known facts, including the above constraints (which we take to be the meaning of "being wise" in this setting).

Common knowledge is realized procedurally by making c.channel accessible to each new instance of a reasoning situation. It is important to point out that, in PROSIT, each time a situation is made a subchunk or a subsituation of a constant, a situation with that constant as name is created (if it did not exist before). So each assumption is made in a newly created situation. The implementation thus respects the properties of the problem we stated in our analysis.

The clauses of our program speak directly about what facts are supposed to be known by each person (i.e. to hold in his knowledge situation), making for a more natural and straightforward program than McCarthy's predicate logic formalization of the wisemen problem in terms of possible worlds ([McCarthy, 1990]). The partiality of information supported by situations obviates the need to specify what is not known, which he must do along with specifying what is known. This in turn allows us to avoid explicit statements about the passage of time. For the wisemen problem, asserting as common knowledge each response to a query suffices to make the wisemen's knowledge increase with time.

5 Discussion: Common Knowledge for Reasoning

Halpern and Moses prove in [Halpern and Moses, 1990] that it is impossible to attain common knowledge. The reason is that there is no safe communication or (what is basically the same) there are no perfect clocks, i.e. no truly simultaneous access to communication channels.

Yet it appears that humans use common knowledge frequently, from which it follows that we sometimes possess it. Considering the wisemen problem, for example, we see that wise men can reason through arbitrarily many iterations: A knows that B knows that C knows that_____The muddy children problem is an even better example; with *k* muddy children, solving the problem requires it iterations of: ((some person) *knows).* Using the shared (i.e. common knowledge) situation is required in order to reason about common knowledge. No bounded number of iterations on individual knowledge will suffice.

We need common knowledge to solve problems like the wisemen puzzle, but Halpern and Moses showed that it is unattainable. How can we resolve this tension?

Halpern and Moses formalized weaker versions of common knowledge which are attainable in practice, and may suffice for carrying out a number of actions. Here we take a different approach: our solution is to assume safe communication even though safety is not really guaranteed. In practice, people don't question the safe communication assumption when trying to solve the wisemen problem. No one objects, "You didn't tell me that A really heard B answer 'No'."[2]

Knowledge possessed in common by a group is nothing more than appropriately coordinated individual knowledge, viz. knowledge that each member of the group has and that each member knows all have. The latter characteristic means common knowledge requires individual knowledge about group members' knowledge. Establishing the required common knowledge involves the diffi-

[3] We made this assumption explicit in our computational solution.

culties that establishing individual knowledge generally involves. For example, how can you know whether I understand and believe something you tell me?

Despite the difficulty of establishing individual knowledge - including the coordinating knowledge that makes something common knowledge of a group of individuals - human agents often function quite successfully by assuming something to be known and acting on that basis while keeping an eye open for indications that the supposed knowledge is false after all.

In point of fact, human reasoning is not the only domain where common knowledge is relevant; it is equally necessary for human communication using natural language. Clark and Marshall [Clark and Marshall, 1986] show that some uses of the definite article (e.g. referring with "the card" to a card on a table seen by two people who are viewing each other) involves common knowledge under the following conditions.

- triple co-presence
- simultaneity assumption
- attention assumption
- rationality assumption

These are exactly the hypotheses that are implicitly assumed by someone trying to solve the wisemen problem.

We propose to deal in a heuristic way with the problem of common knowledge to support reasoning and logic. Just as communication normally succeeds because a person can ordinarily tell if one of the four necessary conditions fails to be met, so also a reasoning agent can safely assume that what appears to be common knowledge is in fact that - unless one of the necessary conditions for genuinely common knowledge fails.

We note the following parallel. Skeptics doubt people even have much individual knowledge. They argue that the connection between reality and human minds is tenuous at best. However, that need not keep logicians from developing mechanisms for reasoning with knowledge.

Someone who doubts the possibility of common knowledge is skeptical about the possibility of communication. Even though one can never be sure about the security of communication, that shouldn't keep one from working with common knowledge - because humans do in fact use it for reasoning.

6 Conclusion

We have shown how to use common knowledge computationally in solving problems involving cooperation of multiple agents, when common knowledge is available. Although the static, circular representation is theoretically beautiful, it is hard to use in actual problem solving because there is no room for bringing in individual (non-common) knowledge. The procedural representation of common knowledge we presented as an alternative applies readily to solving the three wisemen problem.

We have also shown, even if one can never prove common knowledge has been attained, that assuming it has been attained is often safe and efficacious. The ability to detect fairly reliably when, for instance, one of Clark and Marshall's four conditions enumerated in the preceding section is not met suffices as a guideline for when to assume something is common knowledge. In principle, the problem of when one has individual knowledge is about as difficult.

Acknowledgements

The authors are indebted to Joachim Laubsch and Amnon Ribak for helpful contributions to the solution of the wisemen problem presented here, to Michael Frank for his work on implementing PROSIT and to the anonymous UCAI reviewers for comments and criticism.

We also thank the Center for the Study of Language and Information for support and for a most stimulating working environment. The research reported here was supported in part by a grant from the System Development Foundation to Stanford University.

References

[Barwise, 1989] Jon Barwise. *The Situation in Logic.* CSLI Lecture Notes No. 17, University of Chicago Press, 1989.

[Barwise, 1989a] Jon Barwise. 'On the model theory of common knowledge'. In [Barwise, 1989], pp. 201-220.

[Clark and Marshall, 1986] Herbert II. Clark and Catherine R. Marshall. 'Definite reference and mutual knowledge'. In [Joshi *et* al., 1981].

[Devlin, 1991] Keith Devlin. *Logic and Information I: Infons and Situations.* Cambridge University Press, to appear 1991.

[Halpern and Moses, 1990] Joseph Y. Halpern and Yoram Moses. 'Knowledge and common knowledge in a distributed environment'. In *JACM,* 37, 1990, pp. 549-558.

[Joshi *et al*., 1981] Aravind K. Joshi, Bonnie L. Webber and Ivan A. Sag. *Elements of Discourse Understanding.* Cambridge University Press, 1981.

[Levesque *et* al., 1986] Hector J. Levesque, Philip R. Cohen, and Jose H. T. Nunes. 'On acting together'. In *Proc. of AAAI-90,* pp. 94-99, 1990.

[McCarthy, 1990] John McCarthy. 'Formalization of Two Puzzles Involving Knowledge'. In Vladimir Lifschitz (Ed.). *Formalizing Common Sense.* Ablex Publishing Corporation, Norwood NJ, 1990.

[Nakashima *et* al., 1988] Hideyuki Nakashima, Hiroyuki Suzuki, Per-Kristian Halvorsen and Stanley Peters. 'Towards a computational interpretation of situation theory'. In *Proceedings of the International Conference on Fifth Generation Computer Systems, FGCS-88,* Tokyo, 1988.

[Nakashima and Tutiya, 1991] Hideyuki Nakashima and Syun Tutiya. 'Inference *in* a situation *about* situations'. To appear in *Situation Theory and its Applications, 2,* 1991.