Mechanizing epistemic logic with Coq

Pierre Lescanne

Mechanizing epistemic logic with COQ

Pierre Lescanne

Mai 2004

Research Report N^o RR2004-27

Mechanizing epistemic logic with COQ

Pierre Lescanne

Mai 2004

Abstract

I present a mechanization of epistemic logic, also called knowledge logic, I have done using COQ. This work includes a formalization in COQ of epistemic logic and two case studies. This report is an updating of LIP report RR2001-12

Keywords: epistemic logic, mechanizing logic, modal logic

Résumé

Je présente une mécanisation de la logique épistémique, aussi appelée logique de la connaissance que j'ai réalisée en COQ. Ce travail inclue une formalisation en COQ et deux cas d'étude. Ce rapport est une mise à jour du rapport LIP RR2001-12

Mots-clés: logique épistémique, mécanisation de la logique, logique modale

1 Introduction

Epistemic logic is the logic which formalizes *knowledge* of agents. Among many applications it is used in game theories and economic behavior [1], in databases [7] and in verifying cryptographic protocols [8].

I have chosen to embed epistemic logic into the proof assistant COQ [2]. There are many reasons for that. COQ which is based on the *calculus of inductive constructions* offers a very general tool for representing logic theories. In COQ, proofs are objects that are built by a sophisticated computer aided system and exchanged among researchers. Due to lack of space, I cannot fully introduce COQ, but I hope that I give enough information in this paper for a reader to catch much of the concepts necessary to understand the development of epistemic logic presented here.

Shared knowledge, common knowledge. Epistemic logic is also known as the *logic of knowledge*, it deals with *modalities*, which are not part of traditional logic and which modify the meaning of a proposition. For instance such a modality is the *knowledge modality*: "agent Alice knows that ...", written K_{Alice}. There is one knowledge modality K_i for each agent i, so when there are n agents, there are n knowledge modalities. From the K_i's, one can build two new modalities, namely a modality E_g of *shared knowledge*, which modifies a proposition p into a proposition $E_g(p)$ which means that "everyone in the group g knows p" and a modality C_g of *common knowledge*. $C_g(p)$ would say "p is known to everybody in the group g" in a very strong sense since knowledge about p is known at every level of knowledge. Slightly more precisely, if g is the group of agents and p is a proposition, $E_g(p)$ is the conjunction over the $i \in g$ of the $K_i(p)$ and $C_g(p)$ means something like "everybody knows p and everybody knows that everybody knows p and ... and everybody knows that everybody knows that everybody knows ... that everybody knows p..." This infinite conjunction is handled by making $C_g(p)$ a fixpoint. A typical example of common knowledge is *traffic regulation*. When, as a car driver, you enter an intersection you know that the person on your left will let you go, moreover you know that she knows that you have the right to go and you are sure (you know) that she will not go because she knows that you know that she knows that you have the right to go etc. Actually you pass an intersection with a car on your left, because there is a common knowledge between you as a driver and the driver of the other car on the rule of priority. But those who travel have experienced the variability of the common knowledge. Take a *stop sign*. In Europe it means that the person which has a stop sign will let the other to pass the intersection. In some countries, the stop sign is just a decoration of intersections. In the USA, the common knowledge is different since there are intersections of two crossing roads with four stop signs and this has puzzled more than one European.

One main goal of epistemic logic is to handle properly those concepts of *knowledge of an agent*, *shared knowledge* and *common knowledge*. For the reader who wants to know more, the two books [6, 13] are two excellent introductions to epistemic logic. Rules of epistemic logic are given in the appendix.

Deduction rule and presentation à la Hilbert. The well-known *deduction rule* is as follows: if a statement ψ can be deduced from a set Γ of hypotheses augmented by φ, then the theorem "φ implies ψ" can be deduced from Γ

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \Rightarrow \psi}$$

Most of the logics, noticeably the calculus of inductive constructions, fulfill the deduction rule, but modal logic and epistemic logic do not, therefore they cannot be represented directly in COQ. Indeed suppose that we have the deduction rule in epistemic logic (or in modal logic by the way), we would have $p \vdash p$ and then

$$\frac{p \vdash p}{p \vdash K_i\, p}$$

by the *Knowledge Generalization* rule (see appendix) and then $\vdash p \Rightarrow K_i\, p$ using the deduction rule which would translate into *"if p holds then agent i knows p"* which is not what we want in formalizing epistemic logic. Indeed we want to model agents who know part of the truth not all the truth. However in modal logic the following rule is valid

$$\frac{\Gamma \vdash \varphi}{K_i(\Gamma) \vdash K_i(\varphi)}\,\Box$$

where $K_i(\Gamma)$ means that all the propositions ψ in Γ are replaced by $K_i(\psi)$. This extension or a close one is usually used by people who formalize modal logic, but this a drastic extension of natural deduction which does not make it directly embeddable in COQ.

Consequently, for an implementation of epistemic logic one has either to implement the above rule or to formalize propositional logic and predicate calculus in an approach à la Hilbert. Both approaches involves embedding the calculus as a specific theory in COQ. For the approach à la Hilbert, one defines the set of propositions as a `Set` in COQ and the property for a proposition of being a theorem as a predicate on `proposition`. This kind of approach is called *deep embedding* and requires a very expressive logic. The formalization we get eventually is a higher order epistemic logic.

Related works From a mechanical certification point of view, epistemic logic is usually mechanized by model checking [10, 11]. The work presented in this paper (see also [9]) is to our knowledge the first presentation of the mechanization of the proof theory of epistemic logic. Concurrently Paulien de Wind has made her own mechanization using COQ based on an extension of natural deduction with several levels [17] using basically the $\Box$ rule. Notice that epistemic logic is more than modal logic. Not surprisingly there are many attempts to implement modal logic in logical frameworks, the most noticeable one is due to Basin, Matthews and Viganó [3, 4]. See there for a survey of the other approaches. Their implementation is not made in natural deduction, but in a modified natural deduction the so called *labelled modal logic*.

When dealing with examples, we faced the well-known issue discussed at length in books and papers [6, 13, 5] of finding the right modeling for the problem of interest.

The paper is structured as follows. In Section 2, I outline the implementation of predicate calculus in COQ. In Section 3, I describe modal logic and epistemic logic. Section 4 and Section 5 are devoted to two examples. Section 6 summarizes what I learned from this experiment. The whole development in COQ is available on the WEB at `http://perso.ens-lyon.fr/pierre.lescanne/COQ/EPISTEMIC/`.

2 Predicate calculus

If one aims to introduce modal logic, one has to present predicate calculus in a framework à la Hilbert. For this, I introduce a type `proposition` which is an inductive

Set. Its constructors are the implication Imp (written => as an infix), the quantifier Forall (written \-/) and two operators for modalities K and C. An axiomatization for K and C is given in Section 3

```
Inductive proposition: Set :=
    Imp    :  proposition -> proposition -> proposition |
    Forall :  (A:Set) (A -> proposition) -> proposition |
    K      :  nat -> proposition -> proposition         |
    C      :  (list nat) -> proposition -> proposition.
```

I introduce a predicate theorem in the set proposition which tells which propositions are theorems. For instance, (theorem p) (also written |- p) says that proposition p is a theorem in the object theory representing epistemic logic (See appendix A).

Propositional Logic.

First I introduce axioms for intuitionistic logic only. The axioms are therefore just:

```
Hilbert_K: (p,q:proposition) (theorem p => q => p)

Hilbert_S: (p,q,r:proposition)
    (theorem (p => q => r) => (p => q) => p => r)
```

plus the *modus ponens* as a rule:

```
MP: (p,q:proposition) (theorem p => q) -> (theorem p) -> (theorem q).
```

Here -> is the implication in the meta-theory, namely in COQ. In the proposition-as-type approach (a. k. a. Curry-Howard correspondence), -> is also the type constructor for function spaces. Rather naturally, a rule is a way to deduce a new theorem from one or more previous ones and has the form

(theorem Hyp_1) -> $\ldots$(theorem Hyp_n) -> (theorem *Conclusion*).

Classical logic.

Classical logic has been introduced in a separate module by adding the axiom (p:proposition) |- (¬¬p) => p. I put aside classical logic form in order to check for each case whether classical reasoning vs intuitionistic reasoning was needed in proofs or not.

Predicate Logic.

Instead of (Forall A [a:A](P a)) I use the notation \-/ for Forall and a feature of COQ which allows writing \-/[a:A](P a) as COQ rebuilds the A. There are two axioms for universal quantification (see for instance [16] p. 68):

```
Forall1: (A: Set; P:A -> proposition; a:A) (theorem (\-/P) => (P a))

Forall2: (A: Set; P:A -> proposition; q:proposition)
        (theorem (\-/[x:A](q => (P x))) => q => \-/P)
```

and is one rule:

```
ForallRule: (A: Set; P:A->proposition)
    ((x:A)(theorem (P x))) -> (theorem (Forall A P)).
```

The operator (or the quantifier) `Forall` whose signature is part of the definition of `proposition` depends on a set `A` and builds a proposition from a predicate. In our mechanization in COQ a predicate is a function `(A -> proposition)` — notice the use of `->` as a constructor for a function space — and the quantification `Forall` takes a set `A` and a predicate `P` to produce a proposition `(Forall A P)`.

`Forall1` and `Forall12` are two axioms. `Forall1` is the translation in COQ of $\vdash (\forall x : A)P(x) \Rightarrow P(a)$ and `Forall2` is the translation in COQ of $\vdash [(\forall x : A)(q \Rightarrow P(x))] \Rightarrow q \Rightarrow (\forall x : A)P(x)$ provided that x does not occur freely in q.

In `Forall2`, `[x:A] (q => (P x))` represents a predicate which depends on `x`, since `[x:A] (...  x ...  x)` is the function of body `...  x ...  x`. `(x:A)` is the universal meta-quantification over the `A` (i.e., the universal quantification in COQ). Declaring that `p` is a `proposition` (not a function `A->proposition`) means that `p` does not depend on any parameter, in other words neither `x` nor any other variable occurs in `p`. In other words, `p` is just a propositional variable.

`ForallRule` is a rule that says that if for each `x` in `A`, `(P x)` is a theorem, then `(Forall A P)` is a theorem. It translates the rule

$$\frac{\vdash P(x)}{\forall x P(x)}$$

A monadic rule is a statement of the form `|- foo -> |- bar`. A dyadic rule is a statement of the form `|- foo -> |- bar -> |- baz` (see the *modus ponens* above and the *cut rule* below). `ForallRule` sets an interesting connection between the meta-quantification and the quantification in the object theory. This presentation allows us to get rid of the machinery for handling free variables and captures.

Other connectors and quantifiers.

Usually in intuitionistic logic each connector and each quantifier must be defined independently of the others, unlike classical logic where one defines usually only two connectors and one defines the other connectors for those two. In higher order logic, the situation is similar. One can define all the connectors and quantifiers from two of them, even in intuitionistic logic. I use the connector `=>` and the quantifier `Forall` and I derive the other connectors and the quantifier `Exists` from these ones. Notice that `[p,q:proposition]` is equivalent to `[p:proposition][q:proposition]`.

```
Definition And :=
    [p,q:proposition] \-/[r:proposition] (p => q => r) => r.
```

for

$$p \wedge q \triangleq (\forall r : proposition)(p \Rightarrow q \Rightarrow r) \Rightarrow r$$

```
Definition Or :=
    [p,q:proposition] \-/[r:proposition] (p => r) => (q => r) => r).
```

for

$$p \vee q \triangleq (\forall r : proposition)(p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r$$

```
Definition Exist := [A:Set][P: A->proposition]
    \-/[p:proposition] (\-/[a:A]((P a) => p)) => p.
```

for

$$(\exists x : A)(Px) \triangleq (\forall p : proposition)[(\forall a : A)(P(a) \Rightarrow p) \Rightarrow p]$$

In what follows *And* is written `&` and *Or* is written `V`.

Lemmas and derived rules.

For use in later examples, I proved lemmas like

```
Lemma Or_comm: (p,q: proposition) |- (p V q) => (q V p).
```

which says that `V` is commutative. Often in a proof, one wants to reverse the order of the component of a disjunction, for that its companion rule is more convenient:

```
Lemma rule_Or_comm: (p,q: proposition) |- (p V q) -> |- (q V p).
```

When applied, it leads to prove `|- p V q` for goal `|- q V p`. In our experiments, I noticed that the *cut rule* was very handy

```
(p,q,r:proposition) |- p => q -> |- q => r -> |- p => r.
```

This statement is the rule

$$\frac{\vdash p \Rightarrow q \qquad \vdash q \Rightarrow r}{\vdash p \Rightarrow r}$$

which means that to prove `|- p => r` one has to prove a theorem `p => q` and a theorem `q => r`, where `q` is a newly introduction proposition, the *cut* proposition.

Of course, I used it only after proving that it is a derived rule in the system, I mean that I proved the *cut rule* using the axioms and the rules stated in Coq for the logic. Actually its proof comes from the proof of `(p,q,r:proposition) |- (p => q) => (r => p) => r => q` using twice the *modus ponens*.

From the definition of `Exist`, I proved the theorem

$$[(\forall x \in A)(P(x) \Rightarrow q)] \Rightarrow (\exists x \in A)P(x) \Rightarrow q \qquad x \notin FV(q)$$

```
Lemma Exist2: (A: Set)(P:A -> proposition)(q:proposition)
        |- (\-/[x:A]((P x) => q)) => (Exist A P) => q.
```

This way, I found a mistake in the book of Troesltra and van Dalen *Constructivism in mathematics: an introduction* [16]. On page 68 they give the wrong rule $[(\exists x \in A)(P(x) \Rightarrow q)] \Rightarrow (\exists x \in A)P(x) \Rightarrow q$ as a characterization of $\exists$.

3 Modal Logic and Epistemic Logic

Epistemic logic requires knowledge modalities which satisfy the axioms of modal logic. I decided to introduce infinitely many modalities `(K i)` even though most of the examples require only finitely many. In Coq this is an easy task. `K` has the signature `nat -> proposition -> proposition`. From now on, we restrict ourselves to the logic *T*. The other axioms for *S5* are easily introduced. There are two axioms for *T*:

```
Axiom K_K: (i:nat;p,q:proposition) |- (K i p) => (K i p=>q) => (K i q).

Axiom K_T: (i: nat; p:proposition) |- (K i p) => p.
```

and a rule

```
Axiom K_rule: (i: nat; p:proposition) |- p -> |- (K i p).
```

Epistemic logic requires to introduce a modality `E` for *shared knowledge*. This is done in COQ by using the operator `Fixpoint`

```
Fixpoint E [g: (list nat)]: proposition -> proposition :=
        [p:proposition] Cases g of
            nil       =>  TRUE
         | (cons i g1) => (K i p) & (E g1 p)
        end.
```

`E` takes a group `g` of agents and a proposition `p` and returns a proposition `(E g p)`. It is defined *inductively*, i.e., as a fixpoint on the structure of `g`.

$$\begin{array}{rcl} E\ nil\ p & = & TRUE \\ E\ (cons\ i\ g_1)\ p & = & (K\ i\ p)\quad \&\quad (E\ g_1\ p) \end{array}$$

Notice the case when the group is empty. `E` enjoys some nice properties I proved in COQ like

```
(g:(list nat); p,q:proposition) |- ((E g p) & (E g q)) => (E g (p & q)).
```

E satisfies all the axioms of a modality, for instance $E_g(p) \Rightarrow p$, even for the empty group.

C is the modality for *common knowledge*, it is defined by the axiom

```
(g:(list nat);p:proposition) |- (C g p) => (p & (E g (C g p))).
```

and the rule

```
(g:(list nat); p,q:proposition) |- q => (p & (E g q)) -> |- q => (C g p)).
```

i. e.,

$$\frac{}{\vdash C_G(p) \Rightarrow p \wedge E_G(C_G(p))} \qquad \frac{\vdash p \Rightarrow (q \wedge E_G(p))}{\vdash p \Rightarrow C_G(q)}$$

Lemmas about `C`,

we have for instance

```
Lemma C_T: (g:(list nat); p:proposition) |- (C g p) => p.
Lemma C_CE: (g:(list nat); p:proposition) |- (C g p) => (C g (E g p)).
```

i. e., $\vdash C_G(p) \Rightarrow p$ and $\vdash C_G(p) \Rightarrow C_G(E_G(p))$ or a fixpoint property like

```
(g:(list nat); p:proposition) |- (p & (C g (E g p))) => (C g p).
```

i. e., $\vdash p \wedge E_G(C_G(p)) \Rightarrow C_G(p)$

This shows that $C_G(p)$ is a solution of the equation $p \wedge E_G(\varphi) \Leftrightarrow \varphi$ with unknown φ. The rule shows that if ψ is another solution of that equation then $\psi \Rightarrow C_G(p)$.

I took my own axiomatization for E_g and C_G when I realized that my two reference books [6, 13] were wrong in the case of empty sets of agents. More precisely, in the case of an empty group Exercise 3.11 in [6] is wrong: one cannot have $C_\emptyset(\varphi) \Rightarrow \varphi$ since $C_\emptyset(\varphi) \Rightarrow \mathit{True}$. The axioms in [13] are inconsistent, indeed one can easily show that $C_\emptyset(\varphi) \Rightarrow \mathit{True}$ and

$$\text{the rule} \quad \frac{\vdash \varphi}{\vdash C_\emptyset(\varphi)} \quad \text{yields} \quad \frac{\vdash \varphi}{\vdash \mathit{True}}.$$

One may feel that considering an empty set of agents is a logician whim, but in my case it is not, since I based my inductive definition of C_g on the empty set. Moreover one does not want to add the precondition $G \neq \emptyset$ to every statement.

4 The king, the three wise men and the hats

To illustrate the use of our implementation, let us tackle classical examples. The first one is the puzzle of the king, the three wise men and their hats (Figure 1). In [6], Exercise 1.3, it is presented as *"There are three wise men. It is common knowledge that there are three red hats and two white hats. The king puts a hat on the head of each of the three wise men and asks them (sequentially) if they know the color of the hat on their head. The first wise man says that he does not know; the second wise man says that he does not know; then the third man says that he knows"*. In what follows the wise (wo)men are called agents with names Alice, Bob and Carol. Actually in COQ, Alice, Bob and Carol are taken as abbreviations for `(0)`, `(1)` and `(2)`. I weaken the two middle sentences in *"It is a fact that there are red hats and two white hats. The king puts hats on the head of each of the three wise men and asks them (sequentially) if they know whether they wear a white hat on their head."*

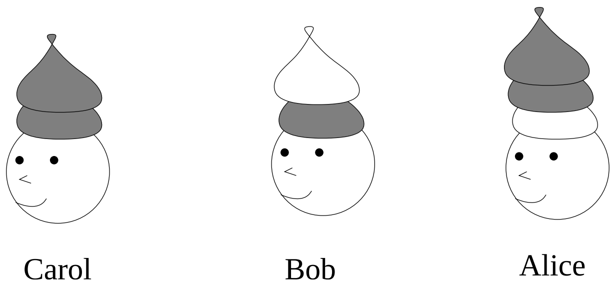

Figure 1: The three wise (wo)men

The puzzle is based on a function

```
Definition Kh := [i:nat] (K i (white i)) V (K i (red i)).
```

which says that the *"agent `i` knows whether or not she (he) wears a white hat"*. With a minimal set of hypotheses, I am able to prove

`|- (K Bob ¬(Kh Alice)) & ¬(Kh Bob) => (red Carol).`

In other words, *"If Bob knows that Alice does not know whether she wears a white hat and if Bob himself does not know whether he wears a white hat, Carol wears only red hats."* If `(red Carol)` is provable from the two premises, then Carol knows that fact; therefore if she knows that *if Bob knows that Alice does not know whether she* *wears a white hat and if Bob himself does not know whether he wears a white hat, Carol wears only red hats*, then she knows that the color of her hats and even more (since she knows that the color of all her hats is red).

The above involved sentences are typical assertions about knowledge. Phrased in English, they are hard to understand for a human. Stated formally they are better understood and they can be checked by a computer

What are the assumptions?

There are five.

- *An agent wears a white hat xor red ones. "xor"* is the exclusive or written `|`.

  ```
   (i:nat) |- (white i) | (red i).
  ```

- *There are only two white hats.* Actually I do not need such a general statement. I only have to state that *"If Bob and Carol wear a white hat, then Alice wears red hats."* which translates in COQ into

  ```
  |- ((white Bob) & (white Carol) => (red Alice)).
  ```

  Note that we are not interested by a statement like "If Carol and Alice wear a white hat, then Bob wears red hats." Moreover the number of red hats is irrelevant and surprisingly an agent can wear more than one hat.

- *Each agent knows the color of the hats of the two other agents.* Actually we are even more restricted than that, namely Alice knows when Bob (resp. Carol) wears a white hat and Bob knows when Carol wears a white hat.

  ```
  |- (white Bob) => (K Alice (white Bob)).
  |- (white Carol) => (K Alice (white Carol)).
  |- (white Carol) => (K Bob (white Carol)).
  ```

  These hypotheses assert that the agents can be supposed to be in a row Carol, Bob, Alice and that each agent knows the color of the hats of the agents before her or him. This is sometime a presentation of this puzzle(see for instance [6] Exercise 1.3 (b)). Actually, I saw in my proof, that the fact that the color of a hat is red is of no interest for any agent.

It should be emphasized that I made actually less hypotheses than in the usual statement of the puzzle.

The proof.

The proof requires just eight small lemmas and needs only modal logic, i. e., no common knowledge. This comes from the fact that "common knowledge" has been replaced by assertion of facts. The mechanization of the proof shows us that many hypotheses made in classical presentation of this puzzle are redundant. Perhaps a careful human analysis of the problem would have lead to the same hypotheses, but what is interesting in this experiment is that this comes naturally from the mechanical development of the proof. One makes the proof and then one traces the hypotheses one actually uses. For instance, in a first attempt we made much more statements about the knowledge of the agents about the color of the hat of the other agents than actually needed. Afterwards, in cleaning up the proof I removed the useless hypotheses leading to the weakening of the initial statement.

The main lemmas are

$\vdash (white\ Bob)\ \&\ (white\ Carol) \Rightarrow (K\ Alice\ (red\ Alice))$.

$\vdash \neg((white\ Bob)\ \&\ (white\ Carol)) \Rightarrow (red\ Bob) \vee (red\ Carol)$.

$\vdash \neg(Kh\ Alice) \Rightarrow (red\ Bob) \vee (red\ Carol)$.

$\vdash \neg(Kh\ Alice)\ \&\ \neg(red\ Carol) \Rightarrow (red\ Bob)$.

where the second one requires a classical proof. The final theorem is

$|-(K\ Bob\ \neg(Kh\ Alice))\ \&\ \neg(Kh\ Bob) \Rightarrow (red\ Carol)$.

with the corollaries:

$\vdash (K\ Carol\ (K\ Bob \neg(Kh\ Alice))\ \&\ \neg(Kh\ Bob)) \Rightarrow (K\ Carol\ (red\ Carol))$.

$\vdash (K\ Carol\ (K\ Bob \neg(Kh\ Alice))\ \&\ \neg(Kh\ Bob)) \Rightarrow (Kh\ Carol)$.

The last corollary means *"If Carol knows that Bob knows that Alice does not know whether or not she wears a white hat and Bob does not know whether or not he wears a white not, then Carol knows whether she wears or not a white hat.* If there only one hat on each head then Carol knows that she were a red hat.

5 The muddy children

This problem is considered by Fagin et al. [6] as the illustration of epistemic logic, especially of common knowledge. Let us give the presentation of [13]. *A number, say n, of children are standing in a circle around their father. There are $k(1 \leq k \leq n)$ children with mud on their heads. The children can see each other but they cannot see themselves. In particular, they do not know if they themselves have mud on their heads. ... Father says aloud: "There is at least one child with mud on its head. Will all children who know they have mud on their heads please step forward?"... This procedure is repeated until, after the k-th time Father has asked the same question, all muddy children miraculously step forward.* I propose a proof of the correctness of the puzzle under reasonable and acceptable hypotheses. The main question is "What does it mean to say that the children see each other and what consequences do they draw from what they see?" For us, "the children see" means that

- they know whether the other children have mud on their head,
- they notice the children stepping forward or not.

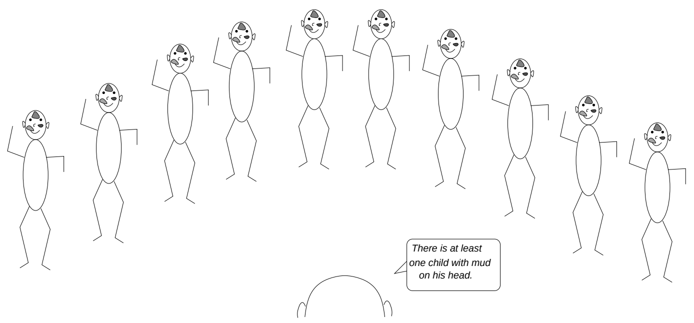

Figure 2: The muddy children

The main interest of the *muddy children* puzzle lies in the use of *common knowledge* (modality `C`).

I define two predicates depending on two naturals, namely `At_least` and `Exactly`. `(At_least n p)` is intended to mean that among the `n` children, there are at least `p` muddy children, whereas `Exactly` means that among the `n` children, there are exactly `p` muddy children. `Exactly` is defined as

```
   [n,p:nat] (At_least n p) &  ¬(At_least n p+1).
```

The hypothesis.

Suppose we are in the situation where

Fact 1 all the children know that there are at least p muddy children

Fact 2 by the fact that none of the children stepped forward, all the children know that there are not exactly p muddy children.

Fact 1 is there since the children know that there is p muddy children because they see them or because they acquired that fact by deduction. Fact 2, namely the knowledge shared by the group `<n>`, which stands for $[0,1,...n]$, on the non exactness of the number p of muddy children, comes from the absence of step forward of children. Therefore Fact 2 is known by every child, formally `K i (E <n> ¬(Exactly n p))`. In other words, after no child has stepped forward, any child knows that all the children know that there are not exactly p children. Therefore I state the axiom:

```
Axiom Knowledge_Diffusion :  (n,p,i:nat)
   |- (E <n> (At_least n p))
       => (E <n> ¬(Exactly n p))
       => (K i (E <n> ¬(Exactly n p))).
```

which is in usually mathematical notation:

$$\vdash E_{\langle n\rangle}(At\_least(n,p)) \Rightarrow E_{\langle n\rangle}(\neg Exactly(n,p)) \Rightarrow K_i(E_{\langle n\rangle}(\neg Exactly(n,p)).$$

This axiom typically describes dynamic in epistemic logic [12]. From it I prove two lemmas:

```
Lemma E_Awareness :  (n,p:nat)
   |- (E <n> (At_least n p))
       => (E <n> ¬(Exactly n p))
       => (E <n> (E <n> ¬(Exactly n p))).
```

i.e., $\vdash E_{\langle n\rangle}(At\_least(n,p)) \Rightarrow E_{\langle n\rangle}(\neg Exactly(n,p)) \Rightarrow E_{\langle n\rangle}(E_{\langle n\rangle}(\neg Exactly(n,p)))$

```
Lemma C_Awareness :  (n,p:nat)
   |- (C <n+1> (At_least n+1 p)) => (E <n+1> ¬(Exactly n+1 p))
                                 => (C <n+1> ¬(Exactly n+1 p)).
```

i.e.,

$$\begin{aligned} \vdash C_{\langle n+1\rangle}(At\_least(n+1,p)) &\Rightarrow E_{\langle n+1\rangle}(\neg Exactly(n+1,p)) \\ &\Rightarrow C_{\langle n+1\rangle}(E_{\langle n+1\rangle}(\neg Exactly(n+1,p))) \end{aligned}$$

Notice that the lemma *C_Awareness* can only be proved for a non empty group of children. I use these lemmas to prove the main result which shows how the knowledge of the children progresses.

```
|- (C <n+1> (At_least n+1 p)) & (E <n+1> ¬(Exactly n+1 p)))
      => (C <n+1> (At_least n+1 p+1))).
```

i. e.,

$$\vdash C_{\langle n+1\rangle}(At\_least(n+1,p)) \quad \wedge \quad E_{\langle n+1\rangle}(\neg Exactly(n+1,p)) \\ \Rightarrow \quad C_{\langle n+1\rangle}(At\_least(n+1,p+1))$$

In other words: *"If it is a common knowledge that there are at least p muddy children and if every child knows that there are not exactly p muddy children then it is a common knowledge that there are at least p+1 muddy children."* Therefore, if for some *k*, a child knows that there is at least $p+1$ muddy children and if he sees p muddy children, he steps forwards. This is the secret of the apparent miracle.

6 What I learned

Do not always trust books I know by experience that what is written in books should be taken with care. But in my case, I found rather fundamental mistakes in my three reference books [16, 6, 13]. This reinforced my claim that *nothing is better than a proof assistant to certify a formalization,* especially on initial and side conditions.

The proof Building proofs in a system à la Hilbert is more difficult for a beginner than in natural deduction as one does not have the ability to discharge hypotheses. Fortunately the use of rules like the *modus ponens*, the *cut rule* or the *rules specific to modal logic and epistemic logic* allows us to organize the proof. One can postpone the proof of some statements of the form `|- ...` and one can divide and conquer proofs. I foresee that some of the tasks of the proof developers can be lightened by tactics to be developed. Anyway, my experience has shown me that after a training the implementation become easy to use.

Acceptable hypotheses A challenge in building proofs in epistemic logic is to state the reasonable and acceptable hypotheses and the acceptable hypotheses are not known a priori. I noticed that I often built proofs of properties backward from the main property I want to prove. Usually there is not so much facility offered by proof assistants, for that. A good approach is to state temporary axioms for the intermediary lemmas and see what can be proved for them and proceed backward, until an acceptable hypothesis is reached.

A didactic tool As said, handling proofs à la Hilbert is a very tedious task when done by hand. You should never do that. This is particularly true in teaching where it is not possible to display huge proof trees in front of a class of students. Therefore I decided to use COQ in lectures dealing with Hilbert presentation of propositional logic. Actually, I give the students (of Ecole normal superieure de Lyon) the first contact with formal logic through the COQ mechanization of propositional logic à la Hilbert. This way, it is easy to tell them the difference between the language et the metalanguage. Propositional logic is the language and COQ is the metalanguage. The students make clearly the difference. Apparently they like this kind of approach and they are quickly able to handle simple proofs in propositional logic in COQ.

Future work I discovered recently the very clever formalization of epistemic logic presented by Masahiko Sato in his PhD [15]. This formalization inspired by John McCarthy uses a super-agent which represents the common knowledge and therefore no fixpoint. I have mechanically proven that his formalization of common knowledge is equivalent to mine, but I did not fully explore all its implications. Anyway it is amazing that such an interesting theory has staid unknown.

Currently René Vestergaard and I investigate the application of my implementation of the epistemic logic to the mechanization Aumann's theorem [1, 14]. Note that in that case, one needs to use epistemic predicate logic,especially Barcan's formula.

References

[1] Robert J. Aumann. Backward induction and common knowledge of rationality. *Games and Economic Behavior*, 8:6–19, 1995.

[2] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, Gérard Huet, Henri Laulhère, César Muñoz, Chetan Murthy, Catherine Parent-Vigouroux, Patrick Loiseleur, Christine Paulin-Mohring, Amokrane Saïbi, and Benjamin Werner. *The Coq Proof Assistant Reference Manual*. INRIA, version 6.3.11 edition, May 2000.

[3] David A. Basin, Seán Matthews, and Luca Viganò. Labelled propositional modal logics: Theory and practic. *J. Log. Comput*, 7(6):685–717, 1997.

[4] David A. Basin, Seán Matthews, and Luca Viganò. Labelled modal logics: Quantifiers. *Journal of Logic, Language and Information*, 7(3):237–263, 1998.

[5] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society*, 426:233–271, 1989.

[6] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[7] Dov M. Gabbay, Christopher John Hogger, and John Alan Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming*, chapter Epistemic and Temporal Logics, Epistemic aspects of databases. Clarendon Press, 1995.

[8] Jon Howell and David Kotz. A formal semantics for SPKI. In *Proceedings of the Sixth European Symposium on Research in Computer Security (ESORICS 2000)*, pages 140–158. Springer-Verlag, October 2000.

[9] Pierre Lescanne. Epistemic logic in higher order logic an experiment with COQ. Technical Report RR2001-12, LIP-ENS de Lyon, 2001.

[10] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Constrcution and Analysis of Systems, TACA'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166, 1996.

[11] Will Marrero, Edmund Clarke, and Somesh Jha. Model checking for security protocols. Technical Report CMU-CS-97-139, Carnegie Mellon University, 1997.

[12] John-Jules Ch. Meyer. *Logic-based artificial intelligence*, chapter Dynamic logic for reasoning about actions and agents, pages 281–311. Kluwer Academic Publishers, 2000.

[13] John-Jules Ch. Meyer and Wiebe van der Hoek. *Epistemic Logic for Computer Science and Artificial Intelligence*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

[14] Dov Samet. Hypothetical knowledge and games with perfect information. *Games and Economic Behavior*, 17:230–251, 1996.

[15] Masahiko Sato. A study of Kripke-type models for some modal logics by Gentzen's sequential method. *Publications of the Research Institute for Mathematical Sciences, Kyoto University*, 13(2):381–468, 1977.

[16] Anne S. Troelstra and Dirk van Dalen. *Constructivism in mathematics: an introduction*, volume 1. North Holland, 1988.

[17] Paulien de Wind. Modal logic in Coq. Master's thesis, Vrije Universiteit Amsterdam, 2002. available at `http://www.cs.vu.nl/~pdwind/thesis/thesis.pdf`.

A The rule of epistemic logic

The epistemic logic has the following axioms and rules. Notice that $\vdash_K \varphi$ means that φ is a classical tautology.

K is sometime called *Distribution Axiom* and **T** is sometime called *Knowledge Axiom*. In our our experiments, we do not use *Introspection* axioms. We give them for information.

$$\frac{\vdash_K \varphi}{\vdash \varphi}\ \textit{Tautologies} \qquad \frac{}{\vdash (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi}\ \mathbf{K}$$

$$\frac{\vdash \varphi \qquad \vdash \varphi \Rightarrow \psi}{\vdash \psi}\ \textit{Modus ponens} \qquad \frac{\vdash \varphi}{\vdash K_i\varphi}\ \textit{Knowledge Generalization}$$

$$\frac{}{\vdash K_i\varphi \Rightarrow \varphi}\ \mathbf{T}$$

$$\frac{}{\vdash K_i\varphi \Rightarrow K_iK_i\varphi}\ \textit{Positive Introspection} \qquad \frac{}{\vdash \neg K_i\varphi \Rightarrow K_i\neg K_i\varphi}\ \textit{Negative Introspection}$$

Figure 3: The basic rules of epistemic logic

Notice that the axiom and the rule given in Figure 4 for C are not the axiom and the rule given in the reference textbooks [6, 13]. We have chosen those ones since they appeared more convenient in the COQ formulation. They are equivalent to the others especially to the ones in [6] for non empty groups of agents. In our axiomatization $C_\emptyset$ satisfies the axioms of modalities namely **K** and **T** since $C_\emptyset(\varphi) \equiv \varphi$ when in [6] $C_\emptyset(\varphi) \equiv$ `TRUE` and $C_\emptyset(\varphi) \Rightarrow \varphi$ does not hold.

One could have added other axioms like *Barcan formula*, which rules the connection between `K` and *Forall* and which can be written readily in COQ:

$$\frac{}{\vdash E_G(\varphi) \Leftrightarrow \bigwedge_{i \in G} K_i \varphi}\ \textit{Definition of E} \qquad \frac{}{\vdash C_G(\varphi) \Rightarrow \varphi \wedge E_G(C_G(\varphi))}\ \textit{Fixpoint}$$

$$\frac{\vdash \varphi \Rightarrow \psi \wedge E_G(\varphi)}{\vdash \varphi \Rightarrow C_G(\psi)}\ \textit{Least Fixpoint}$$

Figure 4: The rules for common knowledge

```
(i: nat) (A:Set) (P:A->proposition) |- (\-/ [x:A](K i (P x))) => (K i (\-/ P))
```

which is

$$(\forall x : A) K_i(P(x)) \Rightarrow K_i((\forall x : A) P(x))$$

and which assumes a strong property on *A*.