Mechanizing common knowledge logic using COQ

9th January 2006 – 18 : 37

Pierre Lescanne

Laboratoire de l'Informatique du Parallélisme,
École Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon 07, FRANCE
E-mail: `Pierre.Lescanne@ens-lyon.fr`

Abstract

This paper proposes a formalization in COQ of common knowledge logic and checks its adequacy on case studies. This exercise allows exploring experimentally the proof-theoretic side of common knowledge logic. This work is original in that nobody has considered higher order common knowledge logic from the point of view of proofs performed on a proof assistant. This work is experimental by nature as it tries to draw conclusions from experiments with a proof assistant.

1 Introduction

We must not judge humans by what they do not know, but by what they know and by the way they know it.

Vauvenargues
Thoughts and Maxims

Epistemic logic is the logic which formalizes *knowledge* of agents. It is an extension of classical (or non classical) logic obtained by adding modalities. Actually one modality is added for each agent: it describes the fact that the agent knows a proposition or a fact. In this paper I am interested in a strong extension of epistemic logic, namely common knowledge logic, which adds a new modality called *common knowledge*. It has been considered first by the philosopher Lewis [22] and later more formally by the economist Aumann [2] (see also the formal presentation of Milgrom [28] where "formal" is not taken at the degree required by a mechanization) . Among many applications it is used in game theory and economic behavior [3, 12], in artificial intelligence [26, 16], in databases [11], in verifying cryptographic protocols [19] and in distributed systems [21].

Since its introduction common knowledge logic has been studied as a logical system, but most the authors [9, 27] consider it from its model-theoretic point of view. In this paper the proof-theoretic aspect and mostly the actual mechanization of the common knowledge logic is considered. Only a few approaches are close to this one. Let me cite Alberucci and Jäger [1], Kaneko [20] and, Lismont [23] but these authors consider only the proof theory aspect whereas I look at how common knowledge logic can actually be implemented in a computer.

What should be expected from mechanizing logic? Unlike human proofs which are sometime fuzzy and lacking on detail, mechanizing proofs shows exactly (in every detail) how theorems are proved from axioms. In particular, when experimenting with a proof assistant it is often the case that weaknesses appear in axiom systems, which could not have been discovered by a careful human examination. Those who have experienced mechanized proof know that bugs are discovered at an amazingly early stage in the process. Most of the time those bugs have to do with limit situations like initializations or "straightforward" or less important cases. This was true in the experiment described in this paper. I do not claim that I found bugs or an inconsistency, but I have shown that axiomatizations proposed in the literature are erroneous. More specifically I have shown that the axiom systems for common knowledge logic proposed by a classical textbook in common knowledge logic is not *robust*. By *"robustness"*, I mean the ability of a system of axioms to stay consistent even when its scope is extended. In the case of a system of axioms for n agents, a natural question is: Is the system still sound when $n = 0$? We will see (Section 4) that this is not a joke in the kind of mechanization which I conducted, but this is an essential feature as it makes full sense to start on 0 an inductive definition on n. I have also shown that often people have difficulties in handling rules and/or common knowledge modalities. Actually in hand made proofs, it is difficult to detect whether a precise implication lies in the theory or in the metatheory. As a consequence, sometimes a proposition is modified by a common knowledge modality because it is associated with a theory implication and sometime it is not because it is associated with a metatheory implication (see Section 6 and Appendix B for a discussion and examples). This confusion between theory and meta-theory happens especially in applications to game theory.

I have chosen to embed common knowledge logic into the COQ proof assistant [4, 8]. 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 particular higher order epistemic logic can be easily implemented. In COQ, proofs are first class citizens, i.e., they are mathematical objects that are built by a sophisticated computer aided system and exchanged among researchers. Currently COQ is used and developed by a large community of users and sophisticated tools are offered. It is clear that the choice of COQ is not a key issue in what follows but its strong logic background makes it really appropriate. However tools like ISABELLE [29] or HOL [13] could have been used.

In what follows, I am not going to fully introduce COQ as it is not the aim of this paper and a good book exists [8], but I hope to give enough information for a reader to catch much of the concepts necessary to understand the development of common knowledge logic presented here. Anyway, the main purpose of this paper is not the use of COQ, but the fact that fully mechanized proofs have been performed and lead to interesting discussions.

Shared knowledge, common knowledge. Common knowledge logic is also known as the *logic of knowledge* [26, 16], 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 a notion of group G of agents and there is one knowledge modality K_i for each agent i in G, 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 φ into a proposition $E_G(\varphi)$ and which means that "everyone in the group G knows φ" and a modality C_G of *common knowledge*. Not all approaches of epistemic logic consider *common knowledge*, but for many people, it is essential for the application and I put a strong emphasis on it. $C_G(\varphi)$ would say "φ is known to everybody in the group G" in a very strong sense since knowledge about φ is known at every level of knowledge. Slightly more precisely, if G is the group of agents and φ is a proposition, $E_G(\varphi)$ is the conjunction over the $i \in G$ of the $K_i(\varphi)$ and $C_G(\varphi)$ means something like "everybody knows φ and everybody knows that everybody knows φ and ... and everybody knows that everybody knows that everybody knows ... that everybody knows φ..." This infinite conjunction is handled by making $C_G(\varphi)$ a fixpoint. For philosophers and economists who, like Aumann [3], study game theory, common knowledge is the basis of *rationality*. A metaphoric example (with the weakness of all metaphor) for common knowledge is *traffic regulations* or more precisely its reification in actual life where drivers are supposed to drive on the right side of roads (common knowledge). When, as a driver, Alice enters an intersection she knows that Bob on her left will let her go, moreover she knows that he knows that she has the right to go and she is sure (she knows) that he will not go because he knows that she knows that he knows that she has the right to go etc. Actually she passes through an intersection with a car on her left, because there is common knowledge on the rule of priority between her as a driver and Bob the driver of the other car. But those who travel have experienced the variability of common knowledge (for instance of the actual implementation of traffic regulations), because common knowledge is a specificity of a group of agents (a country or a community of countries) and changes as the group of agents changes. Actually common knowledge is (a consequence of) the culture or (of) the rationality of a given group of agents. Take a *stop sign*. In Europe it means that the person who has a stop sign will let the other pass through the intersection[1]. In other countries the meaning is different since it is common knowledge among the drivers that nobody will respect the traffic signs and therefore everybody will act appropriately, i.e. nobody will ever assume that the fact that a driver has a stop sign will mean he will let the other pass.

[1]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. Clearly the common knowledge on the meaning of traffi c signs is different between the USA and Europe.

In this paper, the main goal of the implementation of common knowledge logic is to handle properly the concepts of *knowledge of an agent*, *shared knowledge* and *common knowledge*, but also induction and higher order propositions. In COQ it is possible to make assumptions on propositions, like *"there are propositions Alice knows that she knows"*. For the reader who wants to learn more about common knowledge logic, the two textbooks [9, 27] are excellent introductions. Rules of common knowledge logic are given in Section 3.

Deduction rule and Hilbert-style presentation. 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 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) and a rule like *Knowledge Generalization* (Figure 1)

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash K_i(\varphi)}. \qquad \text{We would have } \varphi \vdash \varphi \text{ and then} \qquad \frac{\varphi \vdash \varphi}{\varphi \vdash K_i(\varphi)}$$

and then using the deduction rule we would get $\vdash \varphi \Rightarrow K_i(\varphi)$. This would translate into *"if φ holds then agent i knows φ"* 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

$$\frac{\Gamma \vdash \varphi}{K_i(\Gamma) \vdash K_i(\varphi)}\ (Gen)$$

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 is a drastic extension of natural deduction which does not make it directly embeddable in COQ.

Consequently, for an implementation of common knowledge logic one has either to implement the above rule or to formalize propositional logic and predicate calculus in a Hilbert-style approach. Both approaches involve embedding the calculus as a specific theory in COQ. For the Hilbert style approach, 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 one gets eventually is a higher order common knowledge logic in type theory.

Related works Epistemic logic is usually mechanized by model checking [24, 25]. The work presented in this paper is to my knowledge the first presentation of the mechanization of the proof theory of common knowledge logic based on rules. Concurrently Paulien de Wind has made her own mechanization using COQ based on an extension of natural deduction with several levels [36] using basically the above rule (Gen). It is close to reasoning in Kripke models and no investigation about common knowledge have been conducted. Notice that common knowledge 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 [5, 6]. 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 natural deduction for modal logic*. Sequent calculus and natural deduction for common knowledge logic has been studied by severals authors [1, 20, 23, 32], but none has studied higher order common knowledge logic and none considers mechanization. For instance, common knowledge logic in presence of induction or quantification has not been considered and even less quantification over propositions (see Section 4).

The goal of this paper is to present common knowledge logic, its implementation in COQ and its application to classical examples taken from the literature. This work is experimental by nature. It is structured as follows. In Section 2, I outline the implementation of predicate calculus in COQ. In Section 3, I describe common knowledge logic and, in Section 4, I show how it is implemented. Section 5 and Section 6 are devoted to two examples. Section 7 is the conclusion. The whole development in COQ is available on the WEB at `http://perso.ens-lyon.fr/pierre.lescanne/COQ/epistemic_logic.v`.

2 Predicate calculus in COQ

What is COQ?

Since we are heavily using COQ, it is worthwhile to say a few words about it. For a better and longer introduction, I recommend the book of Pierre Casteran and Yves Bertot "*Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions*" [8]. COQ is a proof assistant that implements a strong logic, based on type theory, namely the calculus of inductive constructions. It considers proof objects as first class citizens. The proof environment helps the user to build proofs using a language adapted for humans, the *vernacular*. When a proof has been built using the proof environment it is a mathematical object, which is assumed to be a proof of a statement and which is given to a proof checker (a reliable algorithm) that checks its correctness, i.e., that checks that it is actually a legal proof of the statement. A proof object can be transmitted to another checker or to another logician. Since the calculus of inductive construction is a powerful logic, implementing an axiomatic system inside COQ is technically not too difficult. Common knowledge logic is *deeply embedded* in COQ, which means that there is no reuse of COQ connectors. This means also that no tactic of the proof assistant can be used to prove theorems of the object logic.

A Hilbert-style presentation

If one aims to introduce modal logic, natural deduction does not work cleanly. A Hilbert-style presentation is required. Here is what one does: one embeds a logic (or a theory) namely common knowledge logic into a metatheory, namely COQ. Both the theory and the metatheory have their implications and their quantifications. The difference will be indicated by syntactical notations. The implication in the theory will be called just the implication when the implication in the metatheory will be called the COQ implication, the same for quantifications.

The set of propositions

First, I introduce a type `proposition` which is an inductive `Set` in COQ. In a rough description, this would give something like

$$p,q : proposition \quad ::= \quad p \Rightarrow q \mid \forall_A \, P \mid K \; i \; p \mid C \; g \; p$$

where $\forall_A$ depends on A, $P : x \mapsto P(x)$ is a function from A to $proposition$[2], i is a natural and, g is a list of naturals. For defining the set *proposition*, COQ uses an inductive definition with four constructors, namely the *implication* `Imp` (later written `==>` as an infix), the *quantifier* `Forall` (later written `\-/`) and two *operators for modalities* `K` and `C`. As noticed, COQ has its own quantification which is used in the definition of `Forall`.

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

In the inductive definition, COQ gives the signature of the constructors[3]. Now we are ready to use abbreviations and I will write `==>` for `Imp` and `\-/` for `Forall`. Notice that the COQ keyword `forall` (with a lower case `f`) is the builtin COQ quantification, whereas our temporary notation `Forall` is the quantification in the object theory.

Once one knows what a proposition is, one can introduce the concept of *theorem*, for that I introduce a predicate `theorem`, abbreviated `|-`, in the set `proposition` which tells which propositions are theorems. For instance, `|- p` says that proposition `p` is a theorem in the object theory representing common knowledge logic[4]. See in appendix A, the full COQ definition of `theorem`.

[2]A quantification applies on a predicate and no variable binding is required a priori.

[3]Technically COQ does not allow using infix abbreviations in definitions, but it allows replacing postfix notations by infix ones. When a new constructor is introduced in an inductive definition, it has to be prefix.

[4]Notice that I do not use the above stated rule about infix and prefix and I write `|- p` instead of the cumbersome *theorem p*, hoping that the reader will forgive me. As the reader understands, this is improper in COQ.

Propositional Logic.

The axioms of propositional logic say that some propositions are the basic theorems of the theory. They use `|-` and they are:

```
Hilbert_K: forall p q : proposition, |- p ==> q ==> p

Hilbert_S: forall p q r : proposition,
       |- (p ==> q ==> r) ==> (p ==> q) ==> p ==> r

Classic_NotNot:  forall p :  proposition, |- (¬¬p) ==> p.
```

plus the *modus ponens* as a rule:

```
MP:  forall p q : proposition, |- p ==> q -> |- p -> |- q.
```

Here `->` is the Coq implication, i.e., the implication in the metatheory. In the propositions-as-types approach (a. k. a. Curry-Howard correspondence), `->` is also the type constructor for function spaces. Notice the notation `fun x:A => ...` for a function from `A` into some other set. A function in `proposition -> proposition` could be the identity written

`fun p:proposition => p.`

A rule in the theory is a way to deduce a new theorem from one or more previous ones. A *n*-adic rule has the form

`|-` Hyp_1 `->` $\ldots$ `|-`Hyp_n `->` `|-` *Conclusion*.

If $n = 0$, it is a logical axiom.

Rule `MP` is used to prove theorems as follows. Suppose that one has a theorem of the form `|-` φ `==>` ψ and a theorem `|-` φ, it suffices to apply `MP` to these two theorems to "produce" the new theorem `|-` ψ. Actually, one is rather in the situation of trying to prove `|-` ψ and one looks for two theorems with can be invoked with `MP` to produce it. For instance, if one has to prove `|- q ==> p ==> p`, one invokes

`apply MP with (p ==> p)`

which produces two subgoals `|- (p ==> p) ==> q ==> p ==> p` (an instance of *Hilbert_K*) and `|- p ==> p`, an already proven meta-theorem.

Predicate Logic.

The syntax As said, instead of `Forall A (fun x:A =>` φ`)`, I write `\-/(fun x:A =>` φ`)` where I use the notation `\-/`. Indeed Coq allows dropping `A` if Coq can infer `A`. It is not needed to bind a variable when this is not necessary, i.e., when the function is not given by an expression, but just by its name. Thus Coq allows writing the shorter notation `\-/P` instead of `\-/(fun x:A => (P x))`. Here I use the Coq notation `=>` in `fun x:A => (f x)` which should not be confused with my notation `==>` for the implication. The below axiom `Forall2` is one of the few exceptions where the two notations `=>` and `==>` are used in the same context.

The axioms There are two axioms for universal quantification (see for instance [33] p. 68):

```
Forall1: forall (A: Set)(P:A -> proposition)(a:A), |- (\-/P) ==> (P a)

Forall2: forall (A: Set)(P: A -> proposition)(q: proposition),
       |- (\-/(fun x:A => (q ==> P x)) ==> q ==> \-/P)
```

and there is one rule:

```
ForallRule: forall (A: Set)(P: A->proposition),
     (forall x:A |- (P x)) -> |- \-/ P.
```

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

`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. Notice how the expression *"provided that x does not occur freely in q"* is taken into account in COQ. Indeed, in `Forall2`, the expression `fun x:A => (q ==> (P x))` represents a predicate which depends on `x`. Declaring that `q` is a `proposition` (not a function in `A->proposition`) means that `q` does not depend on any parameter, in other words neither `x` nor any other variable occurs in `q`, which means that `q` is just a propositional variable.

`ForallRule` is a rule that says that if for each `x` in `A`, `(P x)` is a theorem, then `\-/ P` is a theorem. It translates the monadic rule called $\forall$-*introduction*:

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

In the statement of `ForallRule`, notice the COQ `forall x:A`. Indeed, `ForallRule` establishes an interesting connection between the meta-quantification and the quantification in the object theory. This presentation allows the user to get rid of the machinery for handling free variables and captures, leaving that basic task to COQ. However notice that `ForallRule` requires to use *impredicative* Sets, a feature which is no more accepted by default in the versions of COQ, greater or equal to 8.

Other connectors and quantifiers.

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 from those two. In higher order logic, the situation is similar to classical logic [35]. One can define all the connectors and quantifiers from two of them, even in intuitionistic logic[5]. Hence I use the connector `==>` and the

[5]An axiomatization of higher order intuitionistic epistemic logic is obtained by removing the axiom Classic_NotNot

quantifier `\-/` as primitive and I derive the other connectors namely `AND`, `OR`, `TRUE`, `FALSE` and, `NOT` together with the quantifier `Exists`.

```
Definition AND (p q : proposition) :=
        \-/(fun 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) :=
        \-/(fun 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 FALSE := Forall proposition (fun p : proposition => p).
Definition TRUE := Exist proposition (fun p : proposition => p).
Definition NOT (p : proposition) := p ==> FALSE.
```

for

$$FALSE \triangleq (\forall p : proposition)\ p \qquad TRUE \triangleq (\exists p : proposition)\ p$$

$$\neg p \triangleq p \Rightarrow FALSE.$$

```
Definition Exist (A : Set) (P : A -> proposition) :=
  \-/(fun p:proposition => \-/(fun 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 `&`, *OR* is written `V` and *NOT* is written ¬.

Lemmas and derived rules.

For use in later examples, I proved lemmas like

```
Lemma OR_comm : forall 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 components of a disjunction, for that its companion rule is more convenient:

```
Lemma rule_OR_comm : forall p q : proposition, |- p V q -> |- q V p.
```

It is used as follows. If the goal is `|- q V p`, it boils down to prove `|- p V q`. In my experiments, I noticed that the *Transitivity_of_Imp* rule, namely

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

was very handy. It corresponds to 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.

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

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

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

stated in Coq as

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

The proof relies on a lemma (I called Forall_Imp) that says that for every predicate P_1 and P_2 and every a in A, one has $P_1(a) \Rightarrow [(\forall y \in A)(P_1(y) \Rightarrow P_2(y))] \Rightarrow P_2(a)$, which is a variant of $[(\forall y \in A)(P_1(y) \Rightarrow P_2(y))] \Rightarrow P_1(a) \Rightarrow P_2(a)$, which itself is an instance of `Forall1`. Then one unfolds Exist in the statement of `Exist2`, getting

$$[(\forall x \in A)(P(x) \Rightarrow q)] \Rightarrow (\forall p \in proposition)(((\forall x \in A)P(x) \Rightarrow p) \Rightarrow p) \Rightarrow q$$

Then one applies Forall_Imp with A as *proposition*, a as q, y as p, P_1 as $y \mapsto (\forall x \in A)(P(x) \Rightarrow y)$, then $P_1(q)$ is $(\forall x \in A)(P(x) \Rightarrow q)$ and $P_1(p)$ is $(\forall x \in A)(P(x) \Rightarrow p)$ and P_2 as $x \mapsto x$, then $P_2(y)$ is p and $P_2(a)$ is q. This way, one obtains theorem `Exist2` which should be compared with $[(\exists x \in A)(P(x) \Rightarrow q)] \Rightarrow (\exists x \in A)P(x) \Rightarrow q$ which is the similar rule in [33], on page 68 and was my first attempt of a theorem I was unable to prove.

3 The rules of common knowledge logic

ἐπίστἄμαι: I. *know* how to do, *be able* to do, *capable* of doing.
II. . c. acc., *understand* a matter, *know, be versed* in or *acquainted with*.

Henry George Liddell, Robert Scott,
A Greek-English Lexicon

What is modal logic? Modal logics have been introduced by the philosopher Leibniz in the 17th century. A modal logic logic is a logic in which operators are added to modify the propositions. This can be done to weaken (possibility), to strengthen (necessity), to extend the scope of a proposition over time (temporal logic), to tell the effect of an action on a proposition (dynamic logic) or to assume that an agent knows a fact (epistemic logic) or a group of agents know a fact (common knowledge).

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

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

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

Figure 1: The basic rules of epistemic logic: the system $\mathbb{S}5$

What is common knowledge logic? Common knowledge logic was suggested by the philosopher Lewis in 1969 [22] and formally defined by Aumann [2] in the context of economy (see [12] for an introduction in that context) and further studied in the context of artificial intelligence [16] and computer science [21, 15, 11]. In common knowledge logic the modifiers of proposition, e. g., the modalities are basically of three sorts, namely the *knowledge modality* K_i for each agent i, the *shared modality* E_G for a group G of agents and, the *common knowledge modality* C_G for a group G of agents. Other modalities could be considered but they will not be here. Let me recall what I have mentioned in the introduction, namely that the knowledge modality K_i applied to φ means that agent i "knows" φ, that the shared modality E_G applied to φ means that the group G of agents knows φ and that the common knowledge modality C_G applied to φ means that the shared modality is iterated ad infinitum to φ. As we will see, common knowledge is axiomatized by a fixpoint.

The rule of common knowledge logic In this section, I give the rules of common knowledge logic as they are usually given in the classical literature with no intent to discuss them or propose alternative rules. The terminology changes with the authors, I have taken this of Fagin et al. The common knowledge logic has the axioms and rules given in Figure 1. Notice that $\vdash_K \varphi$ means that φ is a classical tautology. *Knowledge Generalization* is also sometime called *Necessitation.*

K is sometime called *Distribution Axiom* or *Normalization axiom*. It can be also written $\vdash K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\varphi \Rightarrow K_i\psi$ where one sees how K_i acts as a kind of morphism over $\Rightarrow$. These two forms are equivalent as proven in COQ. **T** is sometime called *Knowledge Axiom*. The logic defined by the set *{Tautologies, Modus ponens, Knowledge Generalization, **K**, **T**}* is called $\mathbb{T}$.

Notice that the axiom and the rule given in Figure 2 for C are not the axiom and the rule given in [9]. The difference is in axiom *(Fixpoint)*. I have chosen those ones since they are *robust*, i.e., they stay consistent on a large domain taking the same concept of robustness as this known in software design [34] or statistics [30]. More precisely a robust axiomatization of common knowledge should work even for an empty group of agents. An empty group of agents arises naturally on definitions by induction which are routine in a theorem prover based on type theory like COQ. Indeed, one defines shared knowledge on the empty group of agents first and one extends it by adding one

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

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

Figure 2: The rules for common knowledge

$$\dfrac{C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi\Rightarrow\varphi \qquad \dfrac{\dfrac{C_G(\varphi\Rightarrow E_G(\varphi))\Rightarrow E_G(C_G(\varphi\Rightarrow E_G(\varphi)))\ \boxed{\text{A8}} \qquad \dfrac{C_G(\varphi\Rightarrow E_G(\varphi))\Rightarrow(\varphi\Rightarrow E_G(\varphi))\ \boxed{\text{A7}}}{C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi\Rightarrow E_G(\varphi)}}{C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi\ \Rightarrow\ E_G(C_G(\varphi\Rightarrow E_G(\varphi)))\wedge E_G(\varphi)}}{C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi\ \Rightarrow\ E_G(C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi)}}{\dfrac{\dfrac{C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi\ \Rightarrow\ \varphi\wedge E_G(C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi)}{C_G(\varphi\Rightarrow E_G(\varphi))\wedge\varphi\Rightarrow C_G(\varphi)}\ \textit{Least Fixpoint}}{C_G(\varphi\Rightarrow E_G(\varphi))\Rightarrow\varphi\Rightarrow C_G(\varphi)}}$$

Figure 3: A proof of Meyer and van der Hoek's axiom $(A10)$

agent at a time. C_G (even when G is empty) satisfies the axioms of modalities namely **K** and **T**. Let us look at other systems of axioms and rules.

The axioms of Meyer and van der Hoek. On page 46 of [27] the axioms of common knowledge are

$$\begin{array}{lrcl}
(A6) & E_G(\varphi) & \iff & K_1(\varphi)\wedge\ldots\wedge K_m(\varphi)\\
(A7) & C_G(\varphi) & \Rightarrow & \varphi\\
(A8) & C_G(\varphi) & \Rightarrow & E_G(C_G(\varphi))\\
(A9) & C_G(\varphi)\wedge C_G(\varphi\Rightarrow\psi) & \Rightarrow & C_G(\psi)\\
(A10) & C_G(\varphi\Rightarrow E_G(\varphi)) & \Rightarrow & \varphi\Rightarrow C_G(\varphi)\\
(R3) & \dfrac{\varphi}{C_G(\varphi)} & &
\end{array}$$

This system of axioms is close to ours, as axioms $(A7)$ and $(A8)$ are a splitting of my axiom *Fixpoint* (see appendix B for more detail). Rule $(R3)$ which is a version of rule *Knowledge Generalization* adapted to the modality C_G is easily proved in my system. The main interesting axiom is $(A10)$. $(A10)$ can be proved using COQ in my system of axiom and rule. The COQ proof is sketched in Figure 3. Uses of propositional calculus are assumed and are not shown. Notice that the proof uses $(A7)$ and $(A8)$ which can be proved elsewhere.

Vice-versa, the rule *Least Fixpoint* can be derived in Meyer and van der Hoek's system as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\varphi \Rightarrow \psi \wedge E_G(\varphi)}{\varphi \Rightarrow E_G(\varphi)}}{C_G(\varphi \Rightarrow E_G(\varphi))}\,(R3)}{\varphi \Rightarrow C_G(\varphi)}\,(A10+MP) \qquad \dfrac{\dfrac{\dfrac{\varphi \Rightarrow \psi \wedge E_G(\varphi)}{\varphi \Rightarrow \psi}}{C_G(\varphi \Rightarrow \psi))}\,(R3)}{C_G(\varphi) \Rightarrow C_G(\psi)}\,(A9+MP)}{\varphi \Rightarrow C_G(\psi)}\,(Transitivityof \Rightarrow)$$

Sato's axioms. In [32], Masahiko Sato presents an axiomatization (due to John McCarthy et al. [26]) of common knowledge, which relies on the existence of a specific agent who has the common knowledge. Let us call 0 this agent. It satisfies

$$(\forall i \in G) \vdash K_0(\varphi) \Rightarrow K_i(K_0(\varphi)).$$

Notice that in COQ it is possible to state by a unique statement the above proposition although it is quantified over the set of agents. From this one can deduce $K_0(\varphi) \Rightarrow \varphi \wedge E_G(K_0(\varphi))$ then by *(Least Fixpoint)* $K_0(\varphi) \Rightarrow C_G(\varphi)$. Symmetrically $C_G(\varphi) \Rightarrow K_0(\varphi)$ is a consequence of the lemma `C_T` (see next section). On another hand it is easy to prove that for $i \in G$, $0 \in G$ and $\vdash K_0(\varphi) \Rightarrow C_g(\varphi)$ then $\vdash K_0(\varphi) \Rightarrow K_i(K_0(\varphi))$. Therefore McCarthy et al. axiomatization can be proved in COQ to be equivalent to mine. Actually this axiomatization accepts the group of plain agents to be empty, just take $G = \{0\}$.

The axiom of Fagin et al. The *Fixpoint* axiom given on p. 35 in [9] (see also [1]) is

$$C_G(\varphi) \iff E_G(\varphi \wedge C_G(\varphi))$$

Combined with the definition

$$E_G(\varphi) \iff \bigwedge_{i \in G} K_i(\varphi)$$

this yields $E_\emptyset(\varphi) = true$ which induces $C_\emptyset(\varphi) \iff \mathtt{true}$, in contradiction with Exercise 3.11 which asks to prove $\vdash C_G(\varphi) \Rightarrow \varphi$. Note that on page 67, [9] gives $C_G(\varphi) \Rightarrow E_G(\varphi \wedge C_G(\varphi))$ which yields the formula $C_\emptyset(\varphi) \Rightarrow \mathtt{true}$, i.e., true. The rule, called $(RC1)$, is

$$\dfrac{\varphi \Rightarrow E_G(\psi \wedge \varphi)}{\varphi \Rightarrow C_G(\psi)}\,RC1 \qquad \text{which yields for } \emptyset \qquad \dfrac{true}{\varphi \Rightarrow C_\emptyset(\psi)}$$

In other words, for every φ and every ψ, $\varphi \Rightarrow C_\emptyset(\psi)$ holds. It is fair to say that this book essentially considers models. In this case it is meaningless to speak about an empty set of agents and actually on page 49 line 4, the set of agents is explicitly said to be non empty.

4 Modal Logic and Epistemic Logic in COQ

"Reports that say something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know,"

"We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know."

Defense Secretary of USA Donald Rumsfeld,
at a news briefing in February 2002

Common knowledge logic requires knowledge modalities which satisfy the axioms of modal logic. I introduced infinitely many modalities `(K i)` (see Section 2) even though most of the examples require only finitely many ones. Indeed in COQ this is an easy task because the Set `nat` is given as a primitive. Therefore, `K` has the signature `nat -> proposition -> proposition`. From now on, I restrict myself to the logic $\mathbb{T}$. The other axioms for $\mathbb{S}5$ are easily introduced, but not used in examples. There are two axioms for $\mathbb{T}$:

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

K_T: forall (i: nat) (p:proposition),  |- (K i p)  ==> p.
```

and a rule

```
K_rule: forall (i: nat) (p:proposition), |- p -> |- (K i p).
```

Common knowledge 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) (p : proposition) {struct g} : proposition :=
  match g with
  | nil => TRUE
  | i :: g1 => K i p & E g1 p
  end.
```

`E` is defined by structural induction on the group `g` of agents as said by the annotation `{struct g}`. It takes a proposition `p` and returns a proposition `(E g p)`. `E` defined *by structural induction* on `g` means that

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

Notice the case when the group is empty. `E` enjoys nice properties that can be proved in COQ like

```
 forall (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

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

and the rule

```
forall (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`, one has for instance

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

```
forall (g:list nat) (p:proposition),
        |- (p & (E g (C 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 $p \wedge E_G(\varphi) \iff \varphi$ with unknown φ. The rule shows that if ψ is another solution of that equation then $\psi \Rightarrow C_G(p)$.

Rumsfeld's theorems. To show the power of higher order as implemented in COQ, let me prove three statements due to *Rumsfeld*. These statements are interesting not because of the depth of their proofs, but because like the definition of connectors &, `V` and *Exist* they make quantifications over propositions which are impredicative.

The first statement says that *"we know there are known knowns"*, in other words every agent knows there is a proposition that he knows that he knows.

```
Theorem Rumsfeld1:
  forall i:nat,
     |- K i (Exist proposition (fun p : proposition => K i (K i p))).
```

The proof of this statement requires *Positive Introspection.* The actual known proposition is `TRUE`. The second statement says that *"we know there are known unknowns"* in other words if one considers any agent, he knows there is a proposition that he knows that he does not know.

```
Theorem Rumsfeld2:
   forall i:nat, |- K i (Exist proposition (fun p:proposition => K i
¬(K i p))).
```

The proof of this second statement requires *Negative Introspection.* The actual known proposition is FALSE. The third statement *"there are unknows unknows* means (as expound by its author) there is a proposition φ such that we don't know that we don't know φ, i.e., $(\exists\varphi \in \mathit{proposition})\neg K_i(\neg K_i\varphi)$, which implies $(\exists\varphi \in \mathit{proposition})K_i\ \varphi$, then TRUE is such a proposition.

5 The muddy children

Let us go quickly to the house to clean my head, said Paul.
Countess de Ségur, *Les malheurs de Sophie*

This problem is considered by Fagin et al. [9] as the illustration of common knowledge logic, especially of common knowledge. Let us give the presentation of [27]. *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 me, "the children see" means that

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

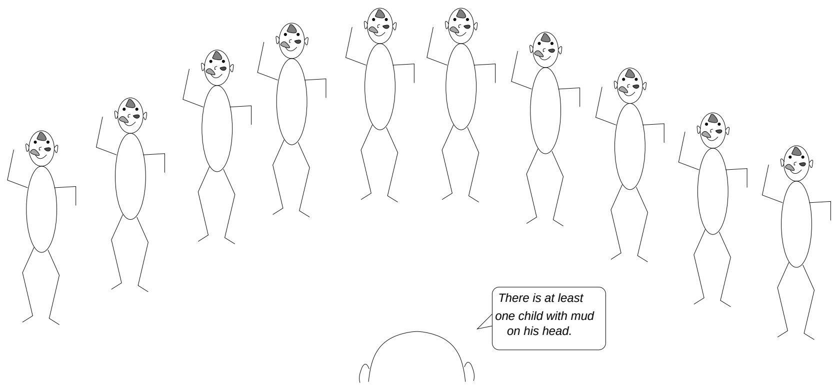

Figure 4: The muddy children

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 (n p : nat)` is defined as `(At_least n p) & ¬(At_least n p+1)`. Moreover `[:n:]` stands for list $[n-1,\ldots 0]$, that is the group of the n children.

The hypothesis.

Suppose that after statements of Father, we have reached a situation where

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

Fact 2 all the children know that there are not exactly p muddy children.

Fact 1 is there since the children know that there are p muddy children because they see them or because they acquired that information by deduction. *Fact 2* is the knowledge shared by the group `[:n:]` on the non exactness of the number p of muddy children. The absence of step forward of children makes *Fact 2* known by every child. Therefore *Fact 2*, namely `(E [:n:] ¬(Exactly n p))`, is known by every child, i.e., `K i (E [:n:] ¬(Exactly n p))`. In other words, after no child has stepped forward, every child knows that all the children know that there are not exactly p children. To summarize at step p,

- *Fact 1* is `E ([:n:]) (At_least n p)`
- *Fact 2* is `(E [:n:] ¬(Exactly n p))`,
- *Conclusion* is `K i (E [:n:] ¬(Exactly n p))`.

and $Fact1 \Rightarrow Fact2 \Rightarrow Conclusion$. In other words, I can state the axiom:

```
Axiom Knowledge_Diffusion : forall 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:

$$\vdash E_{[:n:]}(At\_least(n,p)) \Rightarrow E_{[:n:]}(\neg Exactly(n,p)) \Rightarrow K_i(E_{[:n:]}(\neg Exactly(n,p)).$$

This axiom typically describes dynamic in common knowledge logic [27], Chapter 4. From it, I prove two lemmas:

```
Lemma E_Awareness : forall n p :nat
   |- (E [:n:]  (At_least n p))
       ==> (E [:n:]  ¬(Exactly n p))
       ==> (E [:n:]  (E [:n:]  ¬(Exactly n p))).
```

i.e.,

$$\vdash E_{[:n:]}(At\_least(n,p)) \Rightarrow E_{[:n:]}(\neg Exactly(n,p)) \Rightarrow E_{[:n:]}(E_{[:n:]}(\neg Exactly(n,p)))$$

```
Lemma C_Awareness : forall 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.,

$$\vdash C_{[:n+1:]}(At\_least(n+1,p)) \Rightarrow E_{[:n+1:]}(\neg Exactly(n+1,p)) \Rightarrow C_{[:n+1:]}(\neg Exactly(n+1,p))$$

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, I called `Progress`, which shows how the knowledge of the children progresses.

```
Lemma Progress:
   |- (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.,

$$\vdash C_{[:n+1:]}(At\_least(n+1,p)) \wedge E_{[:n+1:]}(\neg Exactly(n+1,p)) \Rightarrow C_{[:n+1:]}(At\_least(n+1,p+1))$$

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 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.

Discussion

After the above statement, the proof is almost complete, but here I give complements for the interested reader that can be skipped in a first reading.

A lemma on Exactly and At_least. Before starting the proof, a lemma is needed. Assume *Knowledge diffusion* and assume that the children reason perfectly[6]. They should conclude that there is at least `p+1` muddy children, as shown by the following lemma proven in COQ.

```
   Lemma At_least_p_and_not_Exactly_p:  forall n p:nat
       |- (At_least n p) & ¬(Exactly n p) ==> (At_least n p+1).
```

The knowledge diffusion axiom. Here I address one of the main difficulties of using common knowledge logic in practice, namely translating a statement of a scenario (a puzzle or a real live situation) into logical statements. In my case, I have to translate i.e., to formalize the verb "to see" in a formal formula. Dynamic logic [7, 10, 17, 18] can be used for this (see [27] chapter 4). The axiom I propose expresses this.

Now suppose that I am one of these children and that I am the agent *Paul*. After all the previous statements of Father, suppose that everyone knows (shared knowledge) that there is at least `p` muddy children. Moreover suppose that everyone knows (again shared knowledge) that there is not exactly `p` muddy children. Then **I know** by watching the scene that there is not exactly `p` muddy children. This implication *"I see then I know"* is what is meant in the *Knowledge_diffusion*. Thus

[6]"Perfect reasoning" of the agents is a (debatable) assumption of common knowledge logic.

```
Axiom Knowledge_Diffusion_for_Paul :  forall n p:nat
   |- (E [:n:]  (At_least n p))
       ==> (E [:n:]  ¬(Exactly n p))
       ==> (K Paul (E [:n:]  ¬(Exactly n p))).
```

I have taken *Paul* as a generic name but this can be generalized to all the children, hence the universal quantification on `i` (see above). Note that this axiom does not involve any common knowledge.

Why progress in common knowledge and not in shared knowledge? One may wonder why one makes progress in common knowledge and not in shared knowledge. Actually this may work if one would have been able to prove a lemma on the form

$$\vdash E_{[:n:]}(At\_least(n,p)) \Rightarrow E_{[:n:]}(\neg Exactly(n,p)) \Rightarrow E_{[:n:]}(\neg Exactly(n,p))$$

but one is only able to prove a lemma like

$$\vdash E_{[:n:]}(At\_least(n,p)) \Rightarrow E_{[:n:]}(\neg Exactly(n,p)) \Rightarrow E_{[:n:]}(E_{[:n:]}(\neg Exactly(n,p)))$$

with two levels of E in the consequent. This does not allow us to use a generalization rule for E as I was able to do in the proof of

$$\vdash C_{[:n+1:]}(At\_least(n+1,p)) \Rightarrow E_{[:n+1:]}(\neg Exactly(n+1,p)) \Rightarrow C_{[:n+1:]}(\neg Exactly(n+1,p))$$

and this is the key of proof of the *C_Awareness* lemma.

On the strength of common knowledge and on the importance of Father first statement. In common knowledge logic, it is always difficult to acquire a common knowledge. For instance, cryptographic communication through a network relies on the common knowledge of assignments of given public keys to given persons and we know that that these assignments and this common knowledge of the assignments require careful protocols in public key infrastructures. Similarly, in the coordinated attack problem (see [9] section 6.1), generals will be unable to acquire a common knowledge on the agreement for the attack hour on a asynchronous and unreliable network. We also know that teaching traffic regulations on roads requires training and the training is aimed to acquire the common knowledge among the drivers. In our problem, an initial common knowledge is given by Father at the beginning and the lemma I called `Progress` shows how this common knowledge can be enlarged by the other statements that do not involve common knowledge. Without this first statement the kids will not be able to acquire any kind of common knowledge in this respect. They will not be able to increase their common knowledge and even their shared knowledge as well.

Finishing the proof. To complete the proof I consider a given *muddy child* and I try to prove that eventually this muddy child knows there are as many muddy children as children around Father. For that, I declare three variable `nb_children`, `nb_muddy` and `muddy_child` for the total number of children, for the number of muddy children and for a given muddy child. The role of the variable muddy_child is to take one child who has a muddy face and to see that at the end of the process he knows that he has a muddy face. This is can be seen as a kind of Skolemization and take the place of a statement like “at the end, there is child who knows that his face is muddy and therefore steps forwards”. Moreover one needs a few more axioms.

```
Axiom At_least_1:  (le (1) nb_children).
Axiom Muddy_child_is_a_child:  (In muddy_child [:nb_children:]).
Axiom First_Father_Statement:
       |- (C [:nb_children:]  (At_least nb_children (1))).
Axiom What_they_saw:  forall q:nat (lt q nb_muddy) ->
       |- (E [:nb_children:]  ¬(Exactly nb_children q)).
Axiom What_the_muddy_child_sees:
       |- (K muddy_child ¬(At_least nb_children nb_muddy+1)).
```

The first says that there is at least one child. The second one says the muddy child is a child. The third one translates the first Father statement. The fourth one translates what is seen when the children step forward. The last one is what that muddy child sees, that is that there could not be more that nb_muddy+1 muddy children as he (she) sees nb_muddy-1 muddy children.

By induction, one proves

$$\text{If } nb\_muddy > 0 \text{ then } \vdash C_{[:nb\_children:]}(At\_least(nb\_children, nb\_muddy)).$$

If the number of muddy children is greater than 0, then this is a common knowledge that there are nb_muddy muddy children. Then one gets

$$\text{If } nb\_muddy > 0 \text{ then } \vdash K_{muddy\_child}(Exactly(nb\_children, nb\_muddy)).$$

that is *the muddy child knows that there are nb_muddy muddy children.*

6 The king, the three wise men and the hats

So king Solomon exceeded all the kings of the earth for riches and for wisdom.

The First Book of the Kings, X, 23

Is common knowledge needed?

A question that proof theorists ask regularly is whether a given *hypothesis* is actually required in the proof of a given *theorem* in a given *deduction system*. Here the *hypothesis* is common knowledge of fact(s), the *theorem* to prove is the solution of a known and classical puzzle and the *deduction system* is the common knowledge logic as implemented and mechanized in COQ. The question is *“Is common knowledge of the hypotheses required in the proof?”* Since often hand proofs are somewhat sloppy, nothing is better than an implementation to actually verify which statements are used or not used in a proof.

The statement of the puzzle

The classical puzzle I consider is the puzzle of the king, the three wise men and their hats (Figure 5). In [9], 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”.* To ease the reference to them, 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`. In general, the usual assumption is that the statements of the problem are common knowledge among the agents.

The experiments show that no common knowledge is require and in addition I have shown that the two middle sentences can be weaken 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.“*

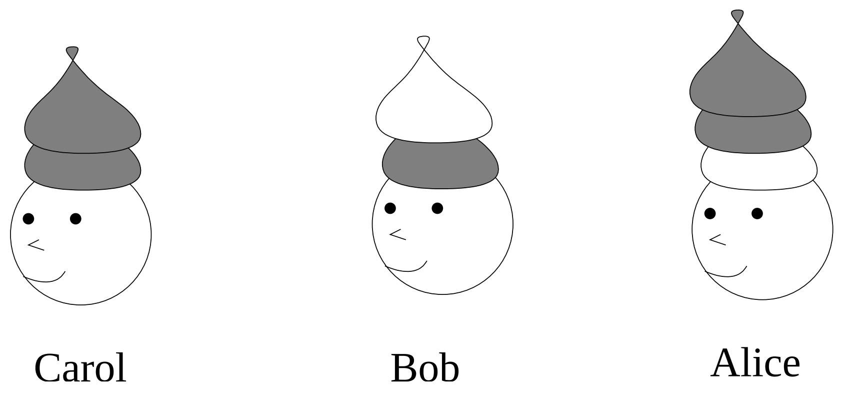

Figure 5: The three wise (wo)men

The puzzle is based on a function

```
Definition Kh := fun i => (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 `|`.

  ```
  forall 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 in 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 (like in Figure 5).

- *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 [9] 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[7].

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

[7]Sato [32] makes the same comment.

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 which are actually used. For instance, in a first attempt I 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 wears a red hat.

Is common knowledge needed after all?

I have chosen to state hypotheses as meta-axioms (axioms in COQ) of the form `|- Facts` and to prove results of the form `|- Result`. Another possibility is to prove statements of the form `|- Hyp ==> Result` in the theory. In that case hypotheses have to be made common knowledge, i.e., `Hyp` is `C [:n:]  Facts`. Indeed asserting a fact as a meta-axiom makes it automatically common knowledge by the rule of *Knowledge Generalization*, in other words if something is a fact it is common knowledge. I am currently experimenting the *king, three wise men and, hats* puzzle along those new lines.

7 Conclusion

Since one cannot be universal and know everything on everything, one must know something on everything. Indeed it is much more beautiful to know something on everything than to know everything on something; this universality is more beautiful.

Blaise Pascal
Pensées

Let me draw some lessons of my experiments

The strength of higher order. COQ supports higher order. Therefore one can state propositions with any kind of quantification, even quantifications over propositions like in Rumsfeld's theorems and one has induction for free. Moreover inductions (induction on natural or structural induction) are builtin and used extensively.

The proofs. Building proofs in a Hilbert-style system is said often to be more difficult than in natural deduction as one does not have the ability to discharge hypotheses. Fortunately the use of rules like *modus ponens, Transitivity_of_Imp* or *rules specific to modal logic and common knowledge logic* allows us to organize the proof. One can postpone the proof of some statements of the form $\vdash \cdots$ 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. The difficulty lies more in understanding epistemic statements.

What logic is needed? The above examples have answered a question one may have when using modal logic, namely: which fragment of logic is required to reason? Our conclusion is that one definitely needs classical logic as a basis. Indeed at some places reasoning based on excluded middle is necessary. On the other hand, except for the noticeable exception of Rumsfeld's theorems, no positive or negative introspection is needed, then $\mathbb{T}$ is enough. Moreover higher order plays a key role in expressing fixpoints and inductions and it is explicitly used in Rumsfeld's theorems where a quantification over propositions is part of the statement.

Right modeling and acceptable hypotheses. A challenge in building proofs in common knowledge logic is to state reasonable and acceptable hypotheses. Unfortunately acceptable hypotheses are not known a priori. I noticed that I built often proofs of properties backward from the conclusion I wanted 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 from them and proceed backward, until an acceptable hypothesis is reached.

Common knowledge vs statements of facts One of the issue in formalizing problems or situation involving common knowledge logic is to choose whether hypotheses have to be written as facts i. e., stated as axioms or added as premises of the implications in the theories. In the second case, they have to be made common knowledge or at least made known by some of the agents. In COQ, we have to choose between `|- Fact ->|- Conclusion` and `|- (C G Fact) ==> Conclusion`. The question of choosing between facts stated as axioms and premises as common knowledge has still to be investigated. Appendix B shows an example.

Future work. Currently René Vestergaard and I investigate the application of my implementation of the common knowledge logic to the mechanization Aumann's theorem [3, 31, 14].

Acknowledgments I acknowledge René Vestergaard for lively discussions on this topic and related ones. I would like to thank Masahiko Sato for pointing me the work done in the context of artificial intelligence in the middle seventies by John Mc Carthy and his coworkers. Daniel Dougherty deserves a special mention for his careful reading and for his interest in this work. The referees gave very accurate comments which contributed to improve the paper greatly.

References

[1] Luca Alberucci and Gerhard Jäger. About cut elimination for logics of common knowledge. *Annals of Pure and Applied Logic*, 2004. to appear.

[2] Robert J. Aumann. Agreeing to disagree. *Annals of Statistics*, 4(6):1236–1239, 1976.

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

[4] 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.

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

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

[7] Mordechai Ben-Ari, Joseph. Y. Halpern, and Amir Pnueli. Deterministic propositional dynamic logic: Finite models, complexity, and completeness. *Journal of Computer and System Sciences*, 25:402–417, 1982.

[8] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development Coq'Art: The Calculus of Inductive Constructions*. Springer-Verlag, 2004.

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

[10] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[11] 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.

[12] John Geanakoplos. *Handbook of Game Theory*, volume 2, chapter *Common knowledge*, pages 1437–1496. Elsevier, Amsterdam, 1994.

[13] Michael J.-C. Gordon and Tom F. Melham. *Introduction to HOL: a theorem proving environment for higher order logic*. Cambridge University Press, 1993. ISBN 0-521-44189-7.

[14] Joseph Y. Halpern. Substantive rationality and backward induction. *Games and Economic Behavior*, 37(2):425–435, 2001.

[15] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 50–61, New York, NY, USA, 1984. ACM Press.

[16] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(3):319–379, 1992.

[17] Joseph Y. Halpern and John H. Reif. The propositional dynamic logic of deterministic, well-structured programs. *Theoretical Computer Science*, 27:127–165, 1983.

[18] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[19] 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.

[20] Mamoru Kaneko. Common knowledge logic and game logic. *The Journal of Symbolic Logic*, 64(2):685–700, June 1999.

[21] Daniel Lehmann. Knowledge, common knowledge and related puzzles (extended summary). In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 62–67, New York, NY, USA, 1984. ACM Press.

[22] David Lewis. *Convention: A philosophical study*. Harvard University Press, Cambridge, MA, 1969.

[23] Luc Lismont. Common knowledge: Relating anti-founded situation semantics to modal logic neighbourhood semantics. *Journal of Logic, Language and Information*, 3:285–302, 1995.

[24] 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.

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

[26] John McCarthy, Masahiko Sato, Takeshi Hayashi, and Shigeru. Igarashi. On the model theory of knowledge. Technical Report AIM-312, Stanford University, 1977.

[27] 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.

[28] Paul Milgrom. An axiomatic characterization of common knowledge. *Econometrica*, 49(1):219–222, 1981.

[29] Larry C. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic in Computer Science*. Academic Press, 1990.

[30] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed.*, chapter *Robust Estimation*, pages 694–700. Cambridge University Press, 1992.

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

[32] 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.

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

[34] Greg Utas. *Robust Communications Software: Extreme Availability, Reliability and Scalability for Carrier-Grade Systems*. John Wiley & Sons, 2005.

[35] Dirk van Dalen. *Logic and Structure*. Springer Verlag, 1994.

[36] 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 COQ definition of *theorem*

```
Inductive theorem :  proposition -> Prop :=

(* ----------------- Propositional calculus ----------------- *)

(* Hilbert axioms for intuitionistic propositional logic *)
| Hilbert_K : forall p q :  proposition, theorem (p ==> q ==> p)
| Hilbert_S : forall p q r :  proposition,
     theorem ((p ==> q ==> r) ==> (p ==> q) ==> p ==> r)
| (* Classic *)
   Classic_NotNot :  forall p :  proposition, theorem (¬¬p ==> p)
```

```
| (* Modus Ponens *) (* p ==> q , p |- q *)
   MP : forall p q :  proposition, theorem (p ==> q) -> theorem p -> theorem
q

(* ------------------ Predicate calculus ------------------ *)

| Forall1 :  forall (A : Set) (P : A -> proposition) (a :  A),
     theorem (\-/P ==> P a)
| (* x not in FV(q) (\-/x (q ==> (P x))) ==> q ==> (\-/x(P x)) *)
   Forall2 :  forall (A : Set) (P : A -> proposition) (q :  proposition),
     theorem (\-/(fun x :  A => (q ==> P x)) ==> q ==> \-/P)

| ForallRule :  forall (A : Set) (P : A -> proposition),
     (forall x :  A, theorem (P x)) -> theorem (\-/P)

(* ----------------- Modal calculus ----------------- *)

| (* Distribution Axiom *)
   K_K : forall (i :  nat) (p q :  proposition),
     theorem (K i p ==> K i (p ==> q) ==> K i q)
| (* Knowledge Axiom *)
   K_T : forall (i :  nat) (p :  proposition), theorem (K i p ==> p)
| (* Knowledge rule *)
   K_rule :  forall (i :  nat) (p :  proposition), theorem p -> theorem (K i
p)
| (* Positive introspection *)
   K_4 :  forall (i :  nat) (p :  proposition), theorem (K i p ==> K i (K i
p))
| (* Positive introspection *)
   K_5 :  forall (i :  nat) (p :  proposition), theorem (¬ K i p ==> K i (¬
K i p))

(* ----------------- Common knowledge logic ----------------- *)

| Fixpoint_C : forall (g :  list nat) (p :  proposition),
     theorem (C g p ==> p & E g (C g p))
| Least_Fixpoint_C : forall (g :  list nat) (p q :  proposition),
     theorem (q ==> p & E g q) -> theorem (q ==> C g p).
```

B More on Meyer and van der Hoek

It is clear that from axiom *(A10)* $C_G(\varphi \Rightarrow E_G(\varphi)) \Rightarrow \varphi \Rightarrow C_G(\varphi)$ and rule *(MP)* we derive

$$\frac{C_G(\varphi \Rightarrow E_G(\varphi))}{\varphi \Rightarrow C_G(\varphi)}$$

and, using rule $(R3)$, a new rule

$$\frac{\varphi \Rightarrow E_G(\varphi)}{\varphi \Rightarrow C_G(\varphi)}\ (nR10)$$

Notice that $(nR10)$ has a flavor of induction for defining C. We could consider rule $(nR10)$ as weaker than axiom $(A10)$, this is not the case. Here is a sketch of the proof of $(A10)$ using $(nR10)$. Let us state $A \ \equiv\ C_G(\varphi \Rightarrow E_G(\varphi))$ in this proof. First, let us prove $A \wedge \varphi \Rightarrow C_G(A \wedge \varphi)$.

$$\dfrac{\dfrac{\dfrac{C_G(\varphi \Rightarrow E_G(\varphi)) \Rightarrow E_G(C_G(\varphi \Rightarrow E_G(\varphi)))\ \boxed{\text{(A8)}}}{A \wedge \varphi \Rightarrow E_G(A)} \qquad \dfrac{\dfrac{C_G(\varphi \Rightarrow E_G(\varphi)) \Rightarrow (\varphi \Rightarrow E_G(\varphi))\ \boxed{\text{(A7)}}}{C_G(\varphi \Rightarrow E_G(\varphi)) \wedge \varphi \Rightarrow (\varphi \Rightarrow E_G(\varphi)) \wedge \varphi} \qquad (\varphi \Rightarrow E_G(\varphi)) \wedge \varphi \ \Rightarrow\ E_G(\varphi)}{C_G(\varphi \Rightarrow E_G(\varphi)) \wedge \varphi \Rightarrow E_G(\varphi)}}{A \wedge \varphi \ \Rightarrow\ E_G(A \wedge \varphi)}}{A \wedge \varphi \Rightarrow C_G(A \wedge \varphi)}\ (nR10)$$

The rest, namely $A \wedge \varphi \Rightarrow C_G(\varphi)$, comes from $(A \wedge \varphi) \Rightarrow \varphi$, then $C_G((A \wedge \varphi) \Rightarrow \varphi)$ and, $C_G(A \wedge \varphi) \Rightarrow C_G(\varphi)$ by $(R3)$, (MP) and transitivity of $\Rightarrow$.

C Barcan's formula.

The axioms of epistemic logic given in Figure 1 are in propositional logic. The *Barcan formula*

$$(\forall P : A \to Bool)\ [(\forall x : A) K_i(P(x))] \Rightarrow K_i((\forall x : A) P(x))$$

is a proposition stated in predicate calculus. It defined the connection between `K` and `\-/`. It can be written readily in Coq:

```
forall (i: nat)(A:Set)(P:A->proposition),
        |- (\-/ fun x:A => K i (P x))  ==> K i (\-/ P)
```

It says that if it occurs that for each element in a set A, if an agent knows a fact, then he knows necessarily that the fact holds for each element in A. This a very strong property which I do not consider except to show that it can be stated easily in Coq.