Dynamic Logic of Common Knowledge in a Proof Assistant

Pierre Lescanne, Jérôme Puisségur

LIP Research Report RR2007-50

Dynamic Logic of Common Knowledge in a Proof Assistant

Pierre Lescanne,* Jérôme Puisségur

Université de Lyon, Ecole Normale Supérieure de Lyon, CNRS (LIP),
46, allée d'Italie, 69364 Lyon 07, FRANCE

Abstract

Common Knowledge Logic is meant to describe situations of the real world where a group of agents is involved. These agents share knowledge and make strong statements on the knowledge of the other agents (the so called *common knowledge*). But as we know, the real world changes and overall information on what is known about the world changes as well. The changes are described by dynamic logic. To describe knowledge changes, dynamic logic should be combined with logic of common knowledge. In this paper we describe experiments which we have made about the integration in a unique framework of common knowledge logic and dynamic logic in the proof assistant COQ. This results in a set of fully checked proofs for readable statements. We describe the framework and how a proof can be conducted.

keywords: Common Knowledge, Dynamic Logic, Proof Assistant

1 Introduction

Common knowledge logic is about the knowledge of the world, whereas *dynamic logic* is about the changes of the world. Both are presented as *modal logic*. In this paper we propose to analyze reasoning in a combination of those logics through a mechanization by a proof assistant.

By experience, we know that the knowledge we have of the world is not perennial, but is meant to evolve. Therefore, any faithful and complete approach of reasoning of agents about their surrounding world requires to take that evolution into account and to combine a logic that describes the state of the knowledge at a given time and a logic that accounts the changes due to external events. This kind of work is known as *belief revision* (or knowledge revision in our case) and is advocated by Johan van

*Corresponding author. email: Pierre.Lescanne@ens-lyon.fr

Benthem [6]. In this paper, following the work of [3, 4, 5, 6, 9], we combine two logics: the first logic is common knowledge logic [1, 12, 17, 20] and the second one is dynamic logic [13, 14]. The combination of both is called *dynamic logic of common knowledge*. The idea is not new but the novelty is that we do that combination in a proof assistant.

As we are neither designers of modal logic, nor philosophers, but only proof assistant users, what is presented in this paper is not a general discussion on the interest or the advantage of combining logics or how this can be made more appropriately. What we present is a record of experiments done on a mechanization of dynamic logic of common knowledge in COQ, one of the proof assistants available on the market. By the use of higher logic and mechanization this activity sheds light on the reality of reasoning in dynamic logic of common knowledge and on how the two components, namely epistemic and dynamic fit together. This paper does not address any comparison on using one proof assistant or another in that kind of implementation exercise. We feel that actually higher order proof assistants like ACL-2 [15], HOL [25], Isabelle [22], LEGO [23], PHoX [24] or PVS [8], are not so deeply different w.r.t. modal logic and that such a comparison would not be informative for the reader. We prefer to focus on the experience itself, hoping that what has been learned will help designers of logics. We have taken COQ, because we practiced it [18] and we have an expert environment around us. This paper is essentially a careful examination of what is necessary to make an actual proof of correctness. We have chosen the *muddy children puzzle* (again not a very original choice) and we introduce the reader to the COQ script.

Why experiences on a proof assistant?

We noticed that most of the presentations about logic of common knowledge or dynamic logic or a combination of both were made either through a model approach where no specific care is given to actual deductions, with rules and axioms[1]. When proofs are given they are done at an intermediary level of abstraction, whereas we advocate a deep level, where no detail is left over. We are typically at a *proof theory* level. With a proof theoretic background, we feel that proofs and deductions are of main importance as it has been shown with most of experience with proof assistants. To summarize, this paper is about the actual integration of common knowledge and dynamic logic in a unique framework in a proof assistant. It relies on a previous work by the first author [18] and is associated with two scripts:

`http://perso.ens-lyon.fr/pierre.lescanne/COQ/EpistemicLogic.v8`

and

`http://perso.ens-lyon.fr/pierre.lescanne/COQ/EpistemicAndDynamicLogic.v`.

[1] A notable exception related to our approach is the formulation of linear temporal logic in COQ done by Solange Coupet-Grimal [7]. Her development is a shallow embedding when ours is a deep one.

2 Dynamic logic of common knowledge

Common knowledge logic

Common knowledge logic is a modal logic with two main modalities. One modality K_i, which is associated with each agent *i*, is the *knowledge modality*. It is meant to express the knowledge an agent has on statements, facts and propositions. For instance, $K_i()$ reads as *i knows* . The modality C_G, which is associated with a group G of agents is the *common knowledge modality*. $C_G()$ translates the fact that a knowledge is common to a group G of agents, not only each agent in the group G knows , but also he knows that the others know and he knows that the others know that the others know , and this recursively. $C_G()$ reads as *is a common knowledge of the group G*. It is formalized as a fixed point by an axiom and a rule:

$$\frac{}{\vdash C_G\varphi \rightarrow \varphi \wedge E_G C_G \varphi}\ FixPoint_C \qquad \frac{\vdash \rho \rightarrow \varphi \wedge E_G \rho}{\vdash \rho \rightarrow C_G \varphi}\ GreatestFixPoint_C$$

Dynamic logic

Dynamic logic makes events modalities. There are as many modalities as there are events. If α is an event, then $[\alpha]$ is a modality and one writes $[\alpha]$ the proposition modified by an event α. If an universe satisfies , after the event α has been performed on it, the transformed universe satisfies $[\alpha]$.

Hilbert-style

Hilbert-style is what has been chosen in the COQ implementation. It is convenient both from the point of view of its presentation and from the point of view of its mechanization in a proof assistant. Therefore the forthcoming rules and axioms will be presented in that framework.

The reason why one cannot use a natural deduction of a sequent calculus approach is essentially due to the Generalization Rule. If one accepts such a rule in natural deduction, one gets

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

This requires to extend the operator K_i to contexts like Γ. If instead of K_i one uses a modality $\Box$, one says that $\Box(\Gamma)$ is a *"boxed context"*. Actually *linear logic* [11] is perhaps the archetypal modal logic and the equivalent of K_i is the modality *of course* written "!". The equivalent of Generalization Rule is a rule called also *of course*. Without that rule the proof net presentation is somewhat simple [16]. Its introduction requires a machinery of boxes which increases its complexity. See [2] for a discussion.

The axioms

The axioms of modal logic are those of *classical logic* plus two axioms and one rule for each modality *M*:

- *Normalization axiom* K_M: $\vdash M\varphi \rightarrow M(\varphi \rightarrow \psi) \rightarrow M\psi$
- *Necessitation axiom* T_M: $\vdash M\varphi \rightarrow \varphi$
- *Generalization rule* Gen_M: $\dfrac{\vdash \varphi}{\vdash M\varphi}$

These axioms of modal logic have to be duplicated for dynamic logic and common knowledge logic.

2.1 Epistemic and dynamic modalities: purely epistemic propositions

The central issue of this paper is to show how to integrate common knowledge and dynamic logics in a unique framework for using in a proof assistant. First we define a logic that we call $\mathcal{T}_{\mathcal{G}}^{\mathcal{C}[\alpha]}$ (see Figure 1). An interesting feature of $\mathcal{T}_{\mathcal{G}}^{\mathcal{C}[\alpha]}$ is axiom **KT1**:

$$\forall : \mathit{proposition}\ \forall \alpha : \mathit{event}\ \forall i \in G, \quad \vdash K_i[\alpha] \rightarrow [\alpha]K_i$$

It is well known in epistemic-temporal logic [10] and is appropriate for dynamic logic of common knowledge. It reads *"if agent i knows that, after event* α*,* φ *holds, then one can infer that, after event* α*, agent i knows that* φ *holds"*. This axiom allows commuting epistemic and dynamic modalities in one direction. Note that the converse is quite dubious in natural language and would certainly be rejected by philosophers. Indeed if *after* α*, I know that holds,* because event α is precisely to let me know proposition , then there no reason to infer that *I know that has to hold after* α. But looking carefully at axiom **KT1**, one notices that event α is transforming not actually the world in its physical reality, but the knowledge the agent has of it. Therefore to avoid troubles and paradoxes, we consider only events α that are *"purely epistemic"*. This means that in our approach of dynamic logic of common knowledge, we consider only actions or events that change the perception of the world which agents have, not the world itself. We borrowed this concept of purely epistemic event from A. Baltag [4, 3].

2.2 The axiomatization of dynamic logic of common knowledge

For the common knowledge modality C_G we have chosen the axiomatization proposed and implemented in Coq by the first of us [18]. The whole dynamic logic of common knowledge is made of the following ingredients:

- the logic $\mathbb{T}$ for K and for $[\alpha]$,
- the definition of *shared knowledge* E_G,
- the definition of *common knowledge* C_G by a fixpoint axiom and a rule that says that it is the greatest fixpoint,
- the axiom **KT1** that makes the connection between dynamic logic and common knowledge logic.

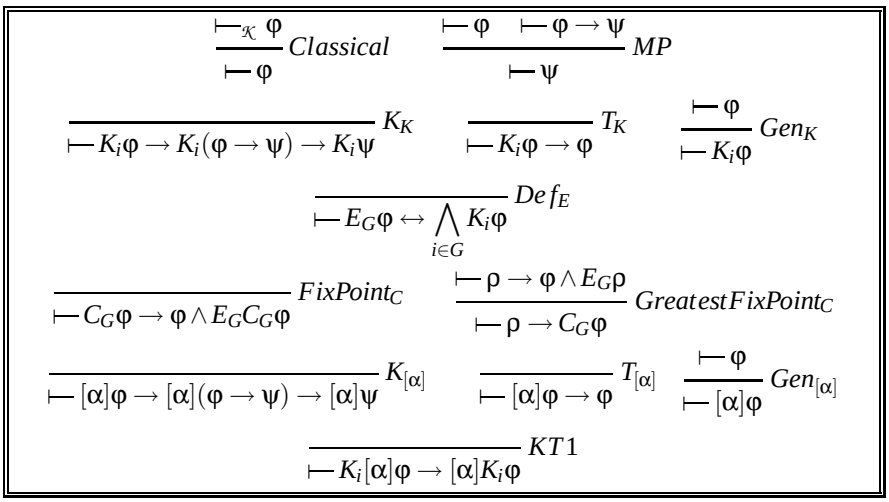

Figure 1: The dynamic logic of common knowledge $\mathcal{T}_G^{C[\alpha]}$

3 A running example: the *muddy children puzzle*

The *muddy children puzzle* will serve as an example to show how dynamic and knowledge logic have been integrated in COQ. This problem is presented by several authors [10, 1, 21] as an illustration of common knowledge logic. The problem considers amazing children who are be able to carry perfectly logical reasoning.

3.1 The statement

First, let us recall the puzzle. The reader who knows the puzzle can skip this part and jump to Section 4, collecting the axioms. We follow more or less the presentation of Meyer and van der Hoek [21].

$n+1$ children are standing in a circle around their father. There are $m+1$ ($m \in \{0,...,c\}$) children with mud on their face. The children can see each other, but they cannot see themselves. In particular, they do not know if they have mud on their face. Father says aloud: "There is at least one child with mud on its face." Then he asks: "Will all children who know they have mud on their face please step forward?" This procedure is repeated until, after the $m+1$-th time Father has asked the same question, all muddy children miraculously step forward.

The conclusion which happens eventually is the result of a logical reasoning made by the children, especially the muddy ones, about what they know initially and what they know about the changes on what they know. It is a perfect example of an common knowledge and dynamic reasoning which fits with our frameworks.

3.2 The formalization

In this section, we try to say what justified our statements. A reader interested only by the formal rules and the mechanized reasoning can jump over the text and go directly to the formal statements. This discussion is interesting to understand why we have chosen this system of axioms.

Two events

In this puzzle, the action are not very elaborated, since after Father's first statement, he keeps repeating the same sentence. Therefore we consider two events, one that starts the scenario and that we write "¤", it is also called the *initial event*, and one that corresponds to the sentence Father repeats and that we will write "*", it is also called the *progression event*. In our dynamic logic of common knowledge, we will have two types of propositions: $[¤]$ and $[*]$. We will also write $[*]^k$ for $[*]...[*]$ where $[*]$ is repeated m times. Clearly $[*]^0$ means . In Coq, we will use the identifiers `Point` (abbreviated in `[¤]` in Coq) and `Star` (abbreviated in `[*]` in Coq).

Definitions

To study this puzzle, we must describe formally the situation and so define basic properties with axioms.

Let $\mathsf{c} \in \mathbb{N}$ and $\mathsf{m} \in \{0,...,\mathsf{c}\}$, so that $\mathsf{c}+1$ is the number of children (there is at least one of them) and $\mathsf{m}+1$ the number of muddy ones (there is also at least one of them). Let G be the group of all children, of cardinality $\mathsf{c}+1$: we identify it with $\{1,...,\mathsf{c}+1\}$.

Let μ_i ($i \in \{1,...,\mathsf{c}+1\}$) be the proposition "child i has mud on his face".

Let λ_j ($j \in \mathbb{N}$) be the proposition "at least j children have mud on their face".

Let ε_j ($j \in \mathbb{N}$) be the proposition "exactly j children have mud on their face", which is defined as follows:

$$\mathbf{EQ}_{\lambda\varepsilon}: \qquad \forall j \in \mathbb{N}, \quad \vdash \varepsilon_j \leftrightarrow \lambda_j \wedge \neg\lambda_{j+1}$$

what one can read "there are exactly j muddy children if and only if there are at least j and at the most j ones". Two trivial properties can be proved from this axiom (the proof is made in the Coq file): first, "if there are at least but not exactly j muddy children, then there are at least $j+1$ ones", which is:

$$\mathbf{IMP}_{\lambda\varepsilon} \qquad \forall j \in \mathbb{N}, \quad \vdash \lambda_j \wedge \neg\varepsilon_j \rightarrow \lambda_{j+1}$$

secondly, a principle of exclusion, "there cannot be exactly j and at least $j+1$ muddy children", which is:

$$\mathbf{EXCLU}_{\lambda\varepsilon} \qquad \forall j \in \mathbb{N}, \quad \vdash \neg(\lambda_{j+1} \wedge \varepsilon_j)$$

These propositions describe the *"physical world"*, i.e., the physical state of the children, whether they are muddy or not. They form the type *physical proposition*. As we only take into account *epistemic events*, physical propositions are *"persistent"*,

which means they are not modified by *epistemic events.* This property is axiomatized as follows:

$$\textbf{PERSIST} \qquad \forall p : \textit{physical proposition}\ \forall \alpha : \textit{epistemic event}, \quad \vdash p \rightarrow [\alpha]p$$

The *initial event* **and its consequences**

First, *Father says loudly that there is at least one muddy child*: therefore this proposition becomes common knowledge. If TRUE is the logical constant, we notice that it is the only "true" proposition available to the children initially. The effect of the first statement is as follows:

$$\textbf{MC1}_1 \qquad \vdash [\boxminus]\mathsf{TRUE} \rightarrow C_G\lambda_1$$

this is the first axiom of our formalization.

The children are not blind, they see each other and they get pieces of information from it. The *initial event* records what they get: *every child counts the number of muddy children in front of him/her*. In particular, the muddy ones see m muddy children, thus they get a knowledge about the total number of muddy children, namely m or $\mathsf{m}+1$:

$$\textbf{MC1}_2 \qquad \forall i \in G, \quad \vdash [\boxminus]\mathsf{TRUE} \rightarrow \mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1})$$

Defined that way, the *initial event* is an *epistemic event*: No further action will change the world, only the knowledge the agents own on the world will evolve. Therefore the muddy children problem is a paradigmatic example.

We said that physical propositions are persistent, but they are not the only ones. Indeed, the muddy children are able to remember what they have seen initially, in other words, the part $\mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1})$ of axiom $\textbf{MC1}_2$ is also persistent:

$$\textbf{PERS}_{MC1_2} \quad \forall \alpha : \textit{event}\ \forall i \in G, \vdash (\mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1})) \rightarrow [\alpha](\mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1}))$$

The final statement

The problem gets to its end when the muddy children step forward. This happens when *muddy children know they are muddy*. Formally this is

$$\forall i \in G, \quad \mu_i \rightarrow K_i\mu_i$$

Muddy children are able to infer this statement when they know there are exactly $\mathsf{m}+1$ muddy children: as every muddy child sees m ones (a persistent property), he knows that he is muddy when he knows there are exactly $\mathsf{m}+1$ muddy children, i.e. the m ones he sees plus him/herself. *If a child is muddy and if he knows there are exactly $m+1$ muddy children, then he knows he is muddy.* This leads to the following axiom.

$$\textbf{MC3} \qquad \forall i \in G, \quad \vdash \mu_i \rightarrow K_i\varepsilon_{\mathsf{m}+1} \rightarrow K_i\mu_i$$

The *progression event* and the increase of knowledge

The core of the work consists in clarifying formally what is produced by Father's injunction and how this makes the muddy children's knowledge to grow.

In this scenario, a tempo is given by Father: time is made discrete and is divided into time intervals which every agents (here the children) can distinguish by counting Father's statements. Therefore, these intervals can be numbered as follows:

- First interval starts at Father's declaration and ends at Father's first injunction
- $(i+1)^{th}$ interval goes from i^{th} to $(i+1)^{st}$ injunction.

After $\mathsf{m}+1$ injunctions, every muddy child steps forward, as we will prove it in our system for dynamic logic of common knowledge. To do so, we need to understand better what happens from an interval to another with each Father's injunction. These injunctions do not carry much semantics, but they are important from a dynamic logic point of view: indeed, each injunction gives a tempo and helps every child in his quest of knowledge as it ends the previous interval. Then every child can deduce that no child has stepped forward during the previous interval which means that none has been able to conclude about his state, these increases the amount of information the children have..

Indeed, let us consider the first injunction. In the first interval, $C_G\lambda_1$ holds and two cases occur:

If $\mathsf{m}=0$, the only muddy child can say at once, that he is muddy because he is the only one to see no other muddy child and after Father's first injunction, he steps forward.

If $\mathsf{m}>0$, every child sees at least another muddy child, and so, no one can conclude whether he is muddy or not. Worst, Father's initial statement of λ_1 did not tell them anything they do not know, but the fact that this statement became common knowledge and when no one steps forward at Father's first injunction, every child can infer that no one sees no muddy child, this means that every one sees at least one muddy child. This can only happen if there are at least two muddy children. By an easy reasoning they exclude the case $\mathsf{m}=0$.

To be more formal, *every child knows that every child knows there is at least one muddy child*, which leads the children to the following: *there are at least two muddy children*.

Father's first injunction translates formally into

$$\vdash E_G E_G \lambda_1 \rightarrow [*] E_G \neg \varepsilon_1$$

which generalizes for any injunction:

$$\textbf{MC2} \qquad \forall j \in \{1,...,k\}, \quad \vdash E_G E_G \lambda_j \rightarrow [*] E_G \neg \varepsilon_j$$

which is *if every child knows that every child knows there are at least j muddy children, then after Father's injunction, every child knows there cannot be exactly j ones.*

4 A knowledge gain lemma

One can deduce a knowledge gain lemma which says that *if every child knows that every child knows there are at least j muddy children, then after Father's injunction, every child knows there are at least $j+1$ ones*. Formally

Lemma 1. GainConn $\quad \forall j \in \{1,...,k\}, \quad \vdash E_G E_G \lambda_j \rightarrow [*] E_G \lambda_{j+1}$

Proof. Let $j \in \{1,...,k\}$.

$$\dfrac{\dfrac{}{\vdash E_G E_G \lambda_j \rightarrow [*] E_G \neg \varepsilon_j}\, MC2 \quad \dfrac{\dfrac{}{\vdash E_G E_G \lambda_j \rightarrow E_G \lambda_j}\, T_E \quad \dfrac{\dfrac{}{\vdash \lambda_j \rightarrow [*]\lambda_j}\, PERS}{\vdash E_G \lambda_j \rightarrow [*] E_G \lambda_j}\, EPers}{\vdash E_G E_G \lambda_j \rightarrow [*] E_G \lambda_j}\, Cut}{\dfrac{\vdash E_G E_G \lambda_j \rightarrow [*] E_G \lambda_j \wedge [*] E_G \neg \varepsilon_j}{\dfrac{\vdash E_G E_G \lambda_j \rightarrow [*](E_G \lambda_j \wedge E_G \neg \varepsilon_j)}{\dfrac{\vdash E_G E_G \lambda_j \rightarrow [*] E_G (\lambda_j \wedge \neg \varepsilon_j)}{\vdash E_G E_G \lambda_j \rightarrow [*] E_G \lambda_{j+1}}\, IMP_{\lambda\varepsilon}}\, E/\wedge Dist}\, */\wedge Dist}\, \wedge Intro$$

□

Summary of the proof of the *muddy children puzzle* theorem

A common knowledge induces a nested shared knowledge at any level, the **GainConn** lemma deduced from **MC2** axiom allows us to get a picture of the proof of the *muddy children puzzle* theorem, which we called **Concl** and which states as:

$$\textbf{Concl} \qquad \vdash \forall \mathsf{m} \in \mathbb{N} \;\; \forall i \in G, \quad [\boxminus]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i \mu_i)$$

Indeed, initially, λ_1 is a common knowledge (**MC1**$_1$), so it is as an arbitrarily nested shared knowledge. With each Father's injunction, children are able to make precise their knowledge about the total number of muddy children by dropping one level of their shared knowledge. Therefore, after j injunctions, they know λ_{j+1} by dropping j depths of their shared knowledge. But since initially this knowledge is arbitrarily deeply nested in shared knowledge, after the first m Father's injunctions, every child effectively knows $\lambda_{\mathsf{m}+1}$.

At this point, the muddy children know there are at least $\mathsf{m}+1$ muddy children; so, as they see m ones, they deduce there are exactly $\mathsf{m}+1$ muddy children (**MC1**$_2$) and they know they are muddy themselves (**MC3**). At the $(\mathsf{m}+1)^{st}$ injunction, they will step forward miraculously, as Meyer and van der Hoek say with humor. After our COQ experiments, we would say perfectly logically!

One can notice that **Concl** holds also for $\mathsf{m}=0$. This theorem describes all the scene: "if Father makes its initial statement, then after the m^{th} injunction, the agents who satisfy property μ know they do.".

5 The proof of the *muddy children puzzle* theorem

In this section, we describe the mechanized proof previously summed up in more detail.

Let $\mathsf{c} \in \mathbb{N}$ and $\mathsf{m} \in \{0,...,\mathsf{c}+1\}$.

Lemma 2 (MultGainConn). $\forall c \in \mathbb{N}^* \ \ \forall j \in \{0,...,m\}, \quad \vdash E_G^{\mathsf{C}+1}\lambda_j \rightarrow [*]E_G^{\mathsf{C}}\lambda_{j+1}$

Proof. By induction on $\mathsf{c} \in \mathbb{N}^*$:

- Initialization : $\mathsf{c} = 1$,

$$\dfrac{}{\vdash E_G E_G \lambda_j \rightarrow [*]E_G\lambda_{j+1}}\ GainConn$$

- Heredity : Let $\mathsf{c} \in \mathbb{N}^*$,

$$\dfrac{\dfrac{\dfrac{}{\vdash E_G^{\mathsf{C}+1}\lambda_j \rightarrow [*]E_G^n\lambda_{j+1}}\ \textsf{HYP-REC}}{\vdash E_G E_G^{\mathsf{C}+1}\lambda_j \rightarrow E_G[*]E_G^n\lambda_{j+1}}\ EDist \qquad \dfrac{}{\vdash E_G[*]E_G^n\lambda_{j+1} \rightarrow [*]E_G^{\mathsf{C}+1}\lambda_{j+1}}\ KT1}{\vdash E_G E_G^{\mathsf{C}+1}\lambda_j \rightarrow [*]E_G^{\mathsf{C}+1}\lambda_{j+1}}\ Cut$$

□

Lemma 3 (ComImpPartIt). $\forall c \in \mathbb{N}, \quad \vdash C_G p \rightarrow E_G^{\mathsf{C}} p$

Proof. By induction on $\mathsf{c} \in \mathbb{N}$:

- Initialization : $\mathsf{c} = 0$,

$$\dfrac{\dfrac{}{\vdash C_G p \rightarrow p \wedge E_G C_G p}\ PointFixe_C}{\vdash C_G p \rightarrow p}\ \wedge Elim$$

- Heredity : Let $\mathsf{c} \in \mathbb{N}$,

$$\dfrac{\dfrac{\dfrac{}{\vdash C_G p \rightarrow p \wedge E_G C_G p}\ PointFixe_C}{\vdash C_G p \rightarrow E_G C_G p}\ \wedge Elim \qquad \dfrac{\dfrac{}{\vdash C_G p \rightarrow E_G^n p}\ \textsf{HYP-REC}}{\vdash E_G C_G p \rightarrow E_G^{\mathsf{C}+1} p}\ EDist}{\vdash C_G p \rightarrow E_G^{\mathsf{C}+1} p}\ Cut$$

□

Lemma 4 (PointImpPartIt). $\forall c \in \mathbb{N}^*, \quad \vdash [\square]\textsf{TRUE} \rightarrow E_G^{\mathsf{C}}\lambda_1$

Proof. Let $\mathsf{c} \in \mathbb{N}^*$.

$$\dfrac{\dfrac{}{\vdash [\square]\textsf{TRUE} \rightarrow C\lambda_1}\ MC1_1 \qquad \dfrac{}{\vdash C_G\lambda_1 \rightarrow E_G^{\mathsf{C}}\lambda_1}\ ComImpPartIt}{\vdash [\square]\textsf{TRUE} \rightarrow E_G^{\mathsf{C}}\lambda_1}\ Cut$$

□

Lemma 5 (PointImpProgr). $\forall c \geq k\ \forall j \in \{1,...,k+1\}$,
$\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{j-1} E_G^{\mathsf{c}-j+1}\lambda_j$

Proof. Let $\mathsf{c} \geq k$. By induction on $j \in \{1,...,k+1\}$:

- Initialization : $j = 1$,

$$\frac{}{\vdash [\Box]\mathsf{TRUE} \rightarrow E_G^{\mathsf{c}}\lambda_1}\ \mathit{PointImpPartIt}$$

- Heredity : Let $j \in \{1,...,\mathsf{m}\}$,

$$\dfrac{\dfrac{\dfrac{}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{j-1}E_G^{\mathsf{c}-j+1}\lambda_j}\ \mathsf{HYP\text{-}REC}}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{j-1}E_G E_G^{\mathsf{c}-j}\lambda_j}\ \mathit{Id} \qquad \dfrac{\dfrac{}{\vdash E_G E_G^{\mathsf{c}-j}\lambda_j \rightarrow [*]E_G^{\mathsf{c}-j}\lambda_{j+1}}\ \mathit{MultGainConn}}{\vdash [*]^{j-1}E_G E_G^{\mathsf{c}-j}\lambda_j \rightarrow [*]^j E_G^{\mathsf{c}-j}\lambda_{j+1}}\ (j-1)*\mathit{Dist}}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^j E_G^{\mathsf{c}-j}\lambda_{j+1}}\ \mathit{Cut}$$

□

From those lemma we get the following ones
With $j = \mathsf{m}+1$

Lemma 6 (ResInter$_1$). $\forall c \geq m, \quad \vdash [\Box]\mathsf{TRUE} \rightarrow [*]^m E_G^{\mathsf{c}-m}\lambda_{m+1}$

With $\mathsf{c} = \mathsf{m}+1$

Lemma 7 (ResInter$_2$). $\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^m E_G\lambda_{m+1}$

And the *muddy children puzzle* theorem comes out (almost) easily.

Theorem 8 (Concl). $\vdash \forall m \in \mathbb{N}\ \ \forall i \in G, \quad [\Box]\mathsf{TRUE} \rightarrow [*]^m(\mu_i \rightarrow K_i\mu_i)$

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}E_G\lambda_{\mathsf{m}+1} \wedge (\mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1}))}\ \mathit{ResInter_2\&MC1_2}}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}E_G\lambda_{\mathsf{m}+1} \wedge [*]^{\mathsf{m}}(\mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1}))}\ \mathit{PERS_{MC1_2}}}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(E_G\lambda_{\mathsf{m}+1} \wedge (\mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1})))}\ */\wedge\mathit{Dist}}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(K_i\lambda_{\mathsf{m}+1} \wedge (\mu_i \rightarrow K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1})))}\ (E_G p \rightarrow K_i p)}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i\lambda_{\mathsf{m}+1} \wedge K_i(\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1}))}\ (a \wedge (b \rightarrow c) \rightarrow (b \rightarrow a \wedge c))}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i(\lambda_{\mathsf{m}+1} \wedge (\varepsilon_{\mathsf{m}} \vee \varepsilon_{\mathsf{m}+1})))}\ K/\wedge\mathit{Dist}}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i((\lambda_{\mathsf{m}+1} \wedge \varepsilon_{\mathsf{m}}) \vee (\lambda_{\mathsf{m}+1} \wedge \varepsilon_{\mathsf{m}+1})))}\ \wedge/\vee\mathit{Dist}}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i((\lambda_{\mathsf{m}+1} \wedge \varepsilon_{\mathsf{m}}) \vee \varepsilon_{\mathsf{m}+1}))}\ (\lambda_{\mathsf{m}} \wedge \varepsilon_{\mathsf{m}} \rightarrow \varepsilon_{\mathsf{m}+1})}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i(\bot \vee \varepsilon_{\mathsf{m}+1}))}\ (\mathit{EXCLU}_{\lambda\varepsilon})}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i\varepsilon_{\mathsf{m}+1})}\ (\bot \vee p \rightarrow p)}{\vdash [\Box]\mathsf{TRUE} \rightarrow [*]^{\mathsf{m}}(\mu_i \rightarrow K_i\mu_i)}\ \mathit{MC2}$$

□

6 The dynamic logic of common knowledge in COQ

6.1 Implementation of $\mathcal{T}_G^{C[\alpha]}$ in COQ

See the appendix for few words on COQ. The implementation presented in this paper is based on another implementation, namely this of the Logic of Common Knowledge done by the first author [18] who implemented all the epistemic multi-agent logic with common knowledge (system $\mathcal{T}_G^C$), of which a COQ file is available on the web: `http://perso.ens-lyon.fr/pierre.lescanne/COQ/EpistemicLogic.v8`

This paper comes out with its own COQ file:
`http://perso.ens-lyon.fr/pierre.lescanne/COQ/EpistemicAndDynamicLogic.v`
which implements the whole system $\mathcal{T}_G^{C[\alpha]}$ and a complete proof of the *muddy children puzzle* theorem **Concl**.

6.2 Why this implementation?

The first aim of this implementation was to ensure a reader that the proof is totally checkable. This lead to a proof of nearly 1100 lines of COQ code, where every lemma is the direct translation of the hand-made proof for a maximal legibility. We do not claim that proof are readable as they would be in an English paper, a certain technicality is required for giving all the detail of the proof; however we claim that the statements of the lemmas are easily readable.

As an added value, this implementation allows any future development by adding axioms or new modalities. This makes our work flexible and reusable.

7 Conclusion

The proof theoretic approach we have used in this paper combines easily epistemic and dynamic logics together, thanks to a general epistemic-dynamic axiom (**KT1**). **(KT1)** involves a commutativity between epistemic modality and a dynamic modality. In the current implementation of **(KT1)**, type is not used to check whether the axiom is only invoked on purely epistemic propositions. In a future implementation, we will create a new type *epistemic proposition* on which **(KT1)** can only be invoked.

After manipulating the logical system presented in this paper with the proof assistant COQ, we feel that it is quite simple and intuitive. It only uses axioms and rules from classical logic plus a few additional axioms and rules. Statements can be made in a language close to this of the hand proof.

The dynamic logic of common knowledge is based on knowledge and events. In a formal statement, an event becomes a dynamic modality which transforms a proposition that describes the world before the event into a proposition that describes the world after that event. Said otherwise a dynamic modality transforms properties into others. Here we have limited our work to epistemic events which only transform agent knowledge, but this is not a big restriction, as this is what happens most of the time.

We notice that we had to adapt the system for the specific situation generated by the *muddy children puzzle*. But this is no so different from situation where classical logic

or another system is used. However, conceptual tools or practical tools (for instance implemented in COQ) could be built to ease the task of the person who mechanizes a proof.

Acknowledgments We would like to thank Stéphane Le Roux, for discussion and advice about COQ.

References

[1] Aumann, R., Hart, S., Eds.: *Handbook of Game Theory*, vol. 2, chapter Common knowledge, Elsevier, Amsterdam, 1994, 1437–1496.

[2] Avron, A., Honsell, F., Miculan, M., Paravano, C.: Encoding Modal Logics in Logical Frameworks., *Studia Logica*, **60**(1), 1998, 161–208.

[3] Baltag, A.: A logic of epistemic actions, *Proceedings of the ESSLLI 1999 workshop on Foundations and Applications of Collective Agent-Based Systems* (W. van der Hoek, J.-J. Meyer, C. Witteveen, Eds.), Utrecht University, 1999.

[4] Baltag, A., Moss, L., Solecki, S.: The logic of public announcements, common knowledge and private suspicion, *Proc. of TARK*, Morgan Kaufmann Publishers, 1998.

[5] van Benthem, J.: *Exploring Logical Dynamics*, CLSI Publications, 1996.

[6] van Benthem, J.: Games in Dynamic Epistemic Logic, *Bulletin of Economic Research*, **53**(4), 2001, 219–248.

[7] Coupet-Grimal, S.: An Axiomatization of Linear Logic, *J Logic Computation*, **13**(6), 2003, 801–813.

[8] Crow, J., Owre, S., Rushby, J., Shankar, N., , Srivas, M.: A Tutorial Introduction to PVS, April 1995.

[9] van Ditmarsch, H. P., van der Hoek, W., Kooi, B. P.: Dynamic epistemic logic with assignment., *AAMAS*, 2005.

[10] Fagin, R., Halpern, J. Y., Moses, Y., Vardi, M. Y.: *Reasoning about Knowledge*, The MIT Press, 1995.

[11] Girard, J.-Y.: Linear Logic, *Theoretical Computer Science*, **50**, 1987, 1–102.

[12] Halpern, J. Y., Moses, Y.: Knowledge and common knowledge in a distributed environment, *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, ACM Press, New York, NY, USA, 1984, ISBN 0-89791-143-1.

[13] Harel, D.: *First-Order Dynamic Logic*, vol. 68 of *Lecture Notes in Computer Science*, Springer-Verlag, 1979.

[14] Harel, D., Tiuryn, J., Kozen, D.: *Dynamic Logic*, MIT Press, Cambridge, MA, USA, 2000, ISBN 0262082896.

[15] Kaufmann, M., Moore, J. S., Manolios, P.: *Computer-Aided Reasoning: An Approach*, Kluwer Academic Publishers, Norwell, MA, USA, 2000, ISBN 0792377443.

[16] Lafont, Y.: From proof nets to interaction nets, *Advances in Linear Logic* (J.-Y. Girard, Y. Lafont, L. Regnier, Eds.), Cambridge University Press, 1995.

[17] Lehmann, D.: Knowledge, common knowledge and related puzzles (Extended Summary), *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, ACM Press, New York, NY, USA, 1984, ISBN 0-89791-143-1.

[18] Lescanne, P.: Mechanizing common knowledge logic using COQ, *Annals of Mathematics and Artificial Intelligence*, **48**(1-2), 2006, 15–43.

[19] Levesque, H. J., Lakemeyer, G.: *The Logic of Knowledge Bases*, MIT Press, 2001.

[20] McCarthy, J., Sato, M., Hayashi, T., Igarashi, S.: *On the Model Theory of Knowledge*, Technical Report AIM-312, Stanford University, 1977.

[21] Meyer, J.-J. C., van der Hoek, W.: *Epistemic Logic for Computer Science and Artificial Intelligence*, vol. 41 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 1995.

[22] Nipkow, T., Paulson, L. C., Wenzel, M.: *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, vol. 2283 of *LNCS*, Springer, 2002.

[23] Pollack *et.al.*, A.: The LEGO Proof Assistant, 2001.

[24] Raffalli, C.: The PhoX Proof Assistant, `http://www.lama.univ-savoie.fr/~RAFFALLI/phox.html`, 2005.

[25] Team, H.: The HOL System DESCRIPTION, September 2005, Kananaskis release.

[26] Weisstein, E. W.: Kepler Conjecture, From MathWorld–A Wolfram Web Resource, `http://mathworld.wolfram.com/KeplerConjecture.html`.

What is COQ?

COQ is a proof assistant, i.e., a program which verifies step by step the validity of a mathematical proof given by the user. In logic, it is generally not obvious to follow a hand-made proof and to determine whether it is right or wrong [26]. A proof assistant, such as COQ, becomes a necessary tool if one chooses to be absolutely sure of a result.

Moreover, COQ is a very good means to build proofs. Indeed, managing a proof step by step, as required by a proof assistant, allows us to understand in a very precise way what is done and what has to be done to complete a proof. COQ is also a way to reach a good formalism as it requires from the user to define exactly all what he manipulates.

Excerpts of the Coq script

Here is the statement of the main lemmas and of the last theorem **Concl**.

```
Lemma GainConn :
  forall  (G: list nat) (i j : nat),
    |- E (i::G) (E (i::G) (lambda j)) ==>
        [*] (E (i::G) (lambda (j+1))).

Lemma MultGainConn :
  forall (G: list nat) (m i j : nat),
    |- F ((m+1)+1) (i::G) (lambda j) ==>
        [*] (F (m+1) (i::G) (lambda (j+1))).

Lemma ComImpPartIt :
  forall (p:proposition) (n:nat) (G: list nat),
    |- C G p ==> F n G p.

Lemma PointImpPartIt :
  forall (G:list nat) (m:nat),
    |- [] TRUE ==> F m G (lambda 1).

Lemma PointImpProgr :
  forall (G:list nat) (i j n:nat),
    |- [] TRUE ==> [*]<:j:> (F (n+1) (i::G) (lambda (j+1) ) ).

Lemma Concl :
  forall (G:list nat) (i j m : nat), In i (j::G) ->
    |- [] TRUE ==> [*]<:m:> (muddy i ==> (K i (muddy i) ) ).
```