Modal Linear Logic in Higher Order Logic
An experiment with COQ

Mehrnoosh Sadrzadeh
msadr016@uottawa.ca *

Abstract

The sequent calculus of classical modal linear logic $KDT4_{lin}$ is coded in the higher order logic using the proof assistant *COQ*. The encoding has been done using two-level meta reasoning in *Coq*. $KDT4_{lin}$ has been encoded as an object logic by inductively defining the set of modal linear logic formulas, the sequent relation on lists of these formulas, and some lemmas to work with lists.This modal linear logic has been argued to be a good candidate for epistemic applications. As examples some epistemic problems have been coded and proven in our encoding in *Coq*: :the problem of logical omniscience and an epistemic puzzle: 'King, three wise men and five hats'

1 Introduction

In this paper we present an encoding of the sequent calculus of modal linear logic using the Coq proof assistant. The logic has been developed in [10] by adding $KDT4$ type modalities to classical linear logic [7].It has the special feature of adding modalities to linear logic on top of linear exponentials. This encoding allows us to state and prove theorems of that logic using facilities of Coq. Previous work in encoding intuitionistic linear logic has been done by associating the constructs of Coq together with linear logic proofs [13]. In our encoding we are treating the modal linear logic as an *object logic* and Coq's Calculus of Inductive Constructs (CIC) as the *meta logic*. This methodology and its benefits has been presented in [6]. Following this methodology linear logic formulas and sequent rules have been inductively defined using the set of inductive datatypes of Coq. Modal logic, too, has been previously encoded in Coq following the same approach [9]. Our encoding will also take advantage of Coq's inductive constructs for implementing sequent rules. The special feature of our logic is that first we are encoding a modal linear logic and second our sequents are classical in the sense that we are not limiting ourselves to sequents with single formulas on the right hand side.The encoding has been done in

*Ph.D. Student, Department of Philosophy, University of Ottawa, Ottawa, Ontario, K1N 6N5, CANADA

three steps: (i) defining modal linear logic formulas inductively, (ii) defining modal linear logic sequent rules inductively, and (iii) extending the modality to an indexed modality thus making our encoded logic a multi-modal linear logic. Following the work of Hintikka [8], modal logics have been widely used to reason about the knowledge of agents. It has been argued that modal linear logic is a good candidate for a non-idealized epistemic logic [3]. In order to study the capabilities of this logic in dealing with epistemic applications, we will use our encoding to prove two standard problems of epistemic logic.

In what follows we first discuss our motivations in working with this modal linear logic in Coq. Then we give a brief overview of the modal linear logic we use.Some familiarity with both linear and modal logics is assumed. We then present our encoding of modal linear logic, and give examples of its application to two well-known epistemic problems: the problem of logical omniscience, and the "King, three wise men, and five hats" puzzle.Finally, we will discuss briefly why we chose classical over intuitionistic logic for the epistemic application.

2 Motivation

Reasoning about knowledge has been a central issue in epistemology since Plato defined knowledge as *justified belief*. In the twentieth century, the discussion was renewed by the use of formal logic and modal operators to account for propositional attitudes such as *I know that...*or *I believe that....* Thus one could use the tools of modern logic to study how agents reason about knowledge. This new branch of logic, called epistemic logic, has found applications in computer science and economics since its inception in the 1960s. But this epistemic logic has had to deal with a serious drawback known as the problem of *logical omniscience*: if an agent knows that p and knows that 'p implies q', deductive closure requires that the agent also knows that q. This is obviously not the case for real agents: we do not know all the consequences, for example, of the axioms of elementary arithmetic. Nor would it be true of a computer because the resources necessary for the knowledge of q might not be available to it, if for example calculation involves exponential complexity.

Thus, epistemic logic embodies strong idealizations about the deductive powers of the agents. One way to go about solving this problem is to find what are the causes of these idealizations. Following Girard's analysis leading to Linear Logic [7], logical systems are made of logical rules concerning the connectives and structural rules, some of them, known as *contraction* and *weakening* are to blame for idealizations. These rules are shown below:

$$\textit{Contraction} \quad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ Left \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ Right$$

$$\textit{Weakening} \quad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ Left \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\ Right$$

Contraction enables us to use a formula infinitely many times in a proof. *Weakening*, on the other hand, allows us to bring unused hypothesis into our proofs resulting in having unrelated hypothesis in the proofs. Hence, proofs in a *substructural* logic that has no such structural rules can neither use a hypothesis infinitely many times, nor use an unrelated hypothesis. These logics should be a good candidate to reason about the knowledge of non-idealized agents. As it will be shown, the theorem that deals with the idealization: the problem of logical omniscience, is still provable in such logic. But this substructural version of logical omniscience should not cause us the idealization problems that are faced in classical modal logics. The reason being that our structural rules are under control in such logic, so the proofs in this logic shall not exceed the epistemic capacities of the agents.

In order to study the capabilities of such logics in dealing with epistemic applications, we have chosen to work on a modal linear logic. Linear logic has several more interesting mathematical properties than being a substructural logic. These mathematical properties might also be considered as proper motivations for an epistemic modal logic. In this paper we have only focused on the resource management and substructural properties of linear logic, leaving the study and applications of other properties as a future work . One problem of applying linear logic to epistemic application is the complexity of its proofs. To partly overcome this problem, we have decided to encode it in the Coq proof assistant. The encoding will enable us to state and prove theorems of the modal linear logic using Coq's facilities. To show the capabilities of this encoding in epistemic applications, we will prove two standard theorems of this field using Coq.

3 Modal Linear Logic $KDT4_{lin}$

Linear Logic is a logic introduced by Girard in 1987 [7] as a refinement of classical logic. It is a substructural logic in the sense that it dismisses *contraction* and *weakening* in their original form. A sequent of the form $\Gamma \vdash \Delta$ in linear logic means that resource presented by Γ are to be consumed yielding resources Δ deduced. This makes linear logic a *resource-sensitive* logic. We can also think of the sequent $\Gamma \vdash \Delta$ as a process that consumes the resources Γ to produce the resources Δ. This *resource-sensitive* property of linear logic makes the conjunction and disjunction of classical logic ambiguous. For example We can use $A \wedge B$ both for producing A and also $A \wedge B$ itself(refer to [7] for a more detailed discussion).To overcome these ambiguities, linear logic uses two distinct connectives for each of conjunction and disjunction. They are called multiplicatives. and additives respectively. We write $A \oplus B$ and $A\&B$ for the two connectives for additives, and $A \otimes B$ and $A \parr B$ for the two multiplicatives. Negation is defined by means of the following sequent rules:

$$\text{Negation} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma, A^{\perp} \vdash \Delta}\ Left \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A^{\perp}, \Delta}\ Right$$

The two multiplicatives are De Morgan duals of each other [2]:

$$\text{Times} \quad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta} \otimes L \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma_1 \vdash B, \Delta_1}{\Gamma, \Gamma_1 \vdash A \otimes B, \Delta, \Delta_1} \otimes R$$

$$\text{Par} \quad \frac{\Gamma, A \vdash \Delta \quad \Gamma_1, B \vdash \Delta_1}{\Gamma, \Gamma_1, A \parr B \vdash \Delta, \Delta_1} \parr L \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta} \parr R$$

The same is true for the two additives. Linear implication will be the same as linear deduction and will be denoted by $A \multimap B$. Multiplicatives, additives and linear implication have left and right sequent rules [7]. An infinite resource, i.e. a resource that can be consumed more than once is shown using the linear exponentials and be written as $!A$ and its De Morgan dual $?A$ [2]:

$$\text{Of Course} \quad \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} !L \qquad \frac{!\Gamma \vdash B, ?\Delta}{!\Gamma \vdash !A, ?\Delta} !R$$

$$\text{Why Not} \quad \frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta} ?L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, Delta} ?R$$

Exponentials are used to control the structural rules mentioned before. Linear *contraction* and *weakening* are shown below:

$$\textit{Contraction} \quad \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta} \; Left \qquad \frac{\Gamma \vdash !A, !A, \Delta}{\Gamma \vdash !A, \Delta} \; Right$$

$$\textit{Weakening} \quad \frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta} \; Left \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash !A, \Delta} \; Right$$

Modalities can be added to linear logic in many different ways. An example is the view that considers exponentials as a kind of modality enriched with structural rules [1].These different ways result in different combinations of linear logic with modal logic, e.g. some such combinations and their semantics have been studied in [10].As part of our motivation, we are interested in a combination that keeps exponentials and adds modalities on top of them. The resulting logic $KDT4_{lin}$ has an algebraic semantics , which has been proven to be sound and complete [10]. The BNF of this logic is shown below:

$$A ::= a|!A|?A|K_iA|A \otimes A|A \parr A|A \oplus A|A\&A|A \multimap A|A^{\perp}|1|0|\top| \perp$$

where:

- a is an atomic formula

- $1, \bot, \top$, and 0 are the units for $\otimes, \parr, \&$, and $\oplus$ respectively
- K is the modal operator
- i is a natural number ranging over a denumerably infinite set
- $K_i A$ intuitively means that i knows A.

The modal axioms and rules of the system $KDT4$ or $S4$ á la Hilbert are:

- Axiom K : $K(A \to B) \to (KA \to KB)$
- Axiom D : $KA \to BA$
- Axiom T : $KA \to A$
- Axiom S4 : $KA \to KKA$
- Generalization(Nec.) rule:

$$\frac{A}{KA}\ Nec$$

Not all of these axioms are independant. For instance axiom D is derivable from axiom T. Although deductive systems for modal logic are traditionally presented in Hilbert style, natural deduction and sequent calculus systems exist. Unfortunately these systems do not satisfy all of the nice properties of modal Hilbert style systems [10]. The sequent calculus presenation of modal logic that we use in this paper can be found in [11, 12].

The single modality K can be extended to multi modalities K_i as mentioned in [10]. We will limit ourselves to a logic with only one multi modality K_i as opposed to one with both B_i and K_i. The sequent rules of this system are shown below. None of these sequent rules involve any of the classical connectives, so they can be used as they are in the modal linear logic KDT_{lin}.

$$\textit{TRules} \quad \frac{\Gamma, A \vdash B, \Delta}{K_i\Gamma, K_iA \vdash B, \Delta}\ Left \qquad \frac{\Gamma, A \vdash B}{K_i\Gamma, K_iA \vdash K_iB}\ Right$$

$$\textit{KDRules} \quad \frac{\Gamma, A \vdash 0}{K_i\Gamma, K_iA \vdash 0}\ Left \qquad \frac{\Gamma, A \vdash B}{K_i\Gamma, K_iA \vdash K_iB}\ Right$$

$$\textit{S4Rules} \quad \frac{\Gamma, A \vdash B, \Delta}{\Gamma, K_iA \vdash B, \Delta}\ Left \qquad \frac{K_i\Gamma, K_iA \vdash B}{K_i\Gamma, K_iA \vdash K_iB}\ Right$$

where the modal rules Γ is a multiset of formulas and $K_i\Gamma$ is the multiset $\{K_iA | A \in \Gamma\}$

Note that if we limit ourselves to only one multi modality K_i, axioms D and T in the Hilbert system become the same. Accordingly, the KD and T sequent rules become the same or in more exact terms T implies KD. Thus we remove the KD rules from our system to avoid redundancy. It can be shown that the modal Hilbert-style axioms (in their multi modal version) are derivable from their sequent counterparts. For example to derive axioms K, it suffices to assume $\Gamma \vdash K_i(A \to B)$ and prove $\Gamma \vdash K_iA \to K_iB$. The proof is straightforward:

$$\dfrac{\Gamma \vdash K_i(A \to B) \quad \dfrac{\dfrac{\dfrac{A \vdash A \quad B \vdash B}{A \to B, A \vdash B}\to L}{K_i(A \to B), K_iA \vdash K_iB}TR}{K_i(A \to B) \vdash K_iA \to K_iB}\to R}{\Gamma \vdash K_iA \to K_iB}Cut$$

Nec. rule takes a more general form:

$$\dfrac{\Gamma \vdash A}{K_i\Gamma \vdash K_iA}\ Nec.$$

It can be shown that the Hilbert-style Nec. rule can be deduced from its sequent counterpart as follows, assume $\Gamma \vdash K_iA$ and prove $\Gamma \vdash A$.

$$\dfrac{\Gamma \vdash K_iA \quad \dfrac{A \vdash A}{K_iA \vdash A}\ S4L}{\Gamma \vdash A}\ Cut$$

It has been mentioned in [5, 9] that the combination of the deduction theorem:

$$\dfrac{\Gamma.A \vdash B}{\Gamma \vdash A \to B}\ Ded.$$

with the Nec. rule makes it possible to prove the wrong proposition that if A is true then agent i knows that A.

$$\dfrac{\dfrac{\Gamma, A \vdash A}{\Gamma, A \vdash K_iA}\ Nec.}{\Gamma \vdash A \to K_iA}\ Ded.$$

It is easy to prove that this problem does not occure in $KDT4_{lin}$ because of the limited version of the Nec. rule. For the complete sequent rules of $KDT4_{lin}$ refer to [10].

4 Encoding the Sequent Rules

The logic $KDT4_{lin}$ is encoded in Coq following the two-level meta-reasoning methodology presented in [6].Thus $KDT4_{lin}$ is treated as an *object* logic while the sequent rules are encoded using the Calculus of Inductive Constructs(CIC).

We shall first define the connectives of $KDT4_{lin}$ inductively, then augment the grammar. In the next step all the sequent rules of our logic are defined inductively using the previously defined linear proposition type. Multi-sets are implemented using Coq's *list* type.Thus, we shall have some lemmas to work with lists.

4.1 Encoding Linear and Modal Connectives

Following [13], we define inductively a set of linear logic propositions: *MLinProp*, which stands for *Modal LINear PROPosition*. The smallest formulas of our modal linear logic will be the different cases of induction.The definition is shown below:

```
Inductive MLinProp : Set :=
| Implies : (MLinProp) → (MLinProp) → MLinProp
| Times : (MLinProp) → (MLinProp) → MLinProp
| Par : (MLinProp) → (MLinProp) → MLinProp
| Plus : (MLinProp) → (MLinProp) → MLinProp
| With : (MLinProp) → (MLinProp) → MLinProp
| OfCourse : (MLinProp) → MLinProp
| WhyNot : (MLinProp) → MLinProp
| Box : (nat) → (list MLinProp)→(list MLinProp)
| Negation : (MLinProp) → MLinProp
| One : MLinProp
| Zero : MLinProp
|⊥ : MLinProp
| ⊤ : MLinProp
```

Now we can use our *MLinProp* as a Coq type. We can define variables of this type. For example we can define A and B as modal linear propositions, and D as a list of Modal linear proposition:

```
Variable A, B : MLinProp.  Variable D : (list MLinProp).
```

We can also define predicates over this type. For example red is a 1-ary modal linear predicate:

```
Variable red:  nat → MLinProp.
```

Note that all of the connectives input linear propositions, but the modality Box, which takes as input a list of linear propositions. This is because the modality sequent rules mentioned before. These rules need our modality to operates over a list of formulas rather than a single formula. The modality is also an indexed modality, making our logic a multi-modal linear logic. A multi-modal logic is a logic an indexed modality One of the applications of such a modality is the epistemic application as will be discussed later. In this application we want to be able to reason about the knowledge of a group of

agents. This means that each modality K_i, has to be indexed to express the knowledge of each agent. For example $K_1 D$ intuitively means that *agent one knows that* D i.e. he knows all of the formulas of the list D. The modality operator can be seen as a binary operator with two operands: an integer! and a list of formulas.

Using Coq's syntax definition and pretty-printing facilities, we can give a notation to each of our modal linear connectives. This will allows us to infix and prefix our connectives.The Coq code for Bang, Times, and Box is given below. We are augmenting the grammar rules and give pretty-printing rules to represent Bang as "!", Times as "*", and Box as "K".The reader is assumed to be familiar with the syntax of these Coq commands (see section 6.7.3 and 6.7.4 of Coq Manual).

```
Grammar command command2 :=
OfCourse [‘‘!’’ command2($c)] →[⟨⟨(OfCourse $c)⟩⟩].
Syntax constr level 2:
[⟨⟨(OfCourse $c)⟩⟩]→[‘‘!’’ $c].
Grammar command command6 :=
Times [command5($c1) ‘‘*’’ command6($c2)] → [⟨⟨(Times $c1 $c2)⟩⟩].
Box [command5($c1) ‘‘K’’ command6($c2)] → [⟨⟨(Box $c1 $c2)⟩⟩].
Syntax constr level 6:
PTimes [⟨⟨(Times $c1 $c2)⟩⟩] → [ $c1:L "*" $c2:E ].
Syntax constr level 6:
PBox [⟨⟨(Box $c1 $c2)⟩⟩]→ [ $c1:L "K" $c2:E ].
```

The notation for all of our modal linear connectives is given in the table below for further reference. The full Coq code has been given in the appendix.

Connective	**Symbol**	**Syntax in Coq**	**Example**
Times	$\otimes$	**	$A**B$
Par	$\parr$	%	$A\%\%B$
Plus	$\oplus$	$\oplus$	$A++B$
With	&	&	$A\&B$
Box	K	K	${}_iKD$
OfCourse	!	!	$!A$
Implies	$\multimap$	$\multimap$	$A \multimap B$

4.2 Encoding Linear and Modal Sequent Rules

In the second phase of our encoding, we will implement the sequent calculus of our modal linear logic. The sequent rules are defined inductively. The induction is made on the linear sequent relation $\Gamma \vdash \Delta$. The sequent relation *LinCons* has been represented as a 2-ary function. It takes two arguments as input: the hypothesisΓ and the conclusion Δ. Remember that Γ and Δ are implemented as lists of formulas. These lists together with the exchange and permutation rules will act as multisets. The output of the linear sequent relation *LinCons* is

either true or false and is defined as a Coq proposition *Prop*. The Coq code for *LinCons* is shown below:

```
Inductive LinCons :  (list MLinProp) → (list MLinProp) →
Prop :=
```

The connective "⊢" is defined as a binary operator with a low precedence using the Coq Syntax and pretty-printing commands:

```
Grammar command command9 :=
LinCons [command8($t1) ‘‘⊢’’ command9($t2)] → [⟨⟨(LinCons $t1 $t2)⟩⟩].
Syntax constr level 9:
PLinCons [⟨⟨(LinCons $t1 $t2)⟩⟩]→ [ $t1 ‘‘⊢’’ $t2 ].
```

The axiom and the sequent rules of the modal linear logic will be the cases of the induction. They are added individually. For example the axiom *Identity* is added as follows:

```
Identity :
(A :  MLinProp)
(‘A ⊢ ‘A)
```

The sequents of our system are of the form $D1 \frown {}‘A \vdash D2 \frown {}‘B$, where $D1$ and $D2$ are lists of formulas of type *MLinProp*, and A and B are formulas of the type *MLinProp*. Note that we have lists on both sides of the sequent. In other words our logic is not intuitionistic as opposed to that of [13]. This helps us to encode all the connectives of linear logic including Par '⅋' Following the encoding of [13], two symbols $\frown$ and ‘ are used to work with lists in Coq; $\frown$ is used to concat two lists and ‘ presents a singleton list. For example, $D1 \frown {}‘A$ concatenates two list $D1$ and the singleton A. The empty list will be shown as*Empty*. Logical and structural rules of modal linear logic are added next. These rules are coded using Coq's implication → for deduction. For example the *Cut* rule [2]:

$$\frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2, A \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ Cut$$

is coded as below:

```
| Cut :
(A, B :  MLinProp)(D1, D2, D3, D4 :  (list MLinProp))
((D1 ⊢ D3 ⌢ ‘A) → (D2 ⌢ ‘A ⊢ D4) → (D1 ⌢ D2 ⊢ D3, D4))
```

As examples of logical rules, the Coq code for Par Left and Times Right is shown below:

```
| ParLeft :
(A, B, C1 , C2 :  MLinProp)(D1, D2 , D3,D4 :  (list MLinProp))
((D1 ⌢  ‘A ⊢ ‘ C1 ⌢ D3) → (D2 ⌢  ‘B ⊢ ‘C2 ⌢ D4) → (D1 ⌢ D2 ⌢
‘(A%%B) ⊢ ‘C1 ⌢  ‘C2 ⌢ D3 ⌢ D4))
```

```
|TimesRight :
(A, B : MLinProp)(D1, D2 , D3 , D4: (list MLinProp))
((D1 ⊢ ‘A ^ D3) → (D2 ⊢ ‘B ^ D4) → (D1 ^ D2 ⊢ ‘(A
** B) ^ D3 ^ D4))
```

The modal sequent rules are T, and S4. The different thing about these rules is that the modal operator has two operands: an index i and a list of formulas D. K_iD will be shown as iKD in Coq.For example the T rule below:

$$\frac{\Gamma, A \vdash B}{iK\Gamma, iKA \vdash iKB}\ T$$

will be code as:

```
| TRule :
(i : nat)(A, B : MLinProp)(D : (list MLinProp))
((D ^ ‘A ⊢ ‘B) → (‘(iKD) ^ ‘(iK‘A) ⊢ ‘(iK‘B)))
```

4.3 Some Lemmas to Work with Lists

For the reason of clarity, we have chosen to work with sequents with distinguished formulas to the left and right hand sides of "⊢":

$$\Gamma, A \vdash B, \Delta$$

But most of the time Γ and Δ are empty lists and the sequent is of the form:

$$A \vdash B$$

Sequents of this form cause us problem because encoded rules of our logic cannot be applied to them. As an example consider the following deduction which is a valid one:

$$\frac{A \vdash A}{A \vdash A \oplus B}\ \oplus R1$$

We will face difficulties in proving this deduction because it does not match the general format of the encoded seqeunts. Thus no rule can be applied to $A \vdash A \oplus B$ whereas $\oplus R$ should be applicable. To solve the problem we will have to add Nil lists to the left hand side of the leading formulas A and $A \oplus B$. Thus the above deduction will look like the following after these changes:

$$\frac{\mathrm{E}mpty, A \vdash \mathrm{E}mpty, A}{\mathrm{E}mpty, A \vdash \mathrm{E}mpty, A \oplus B}\ \oplus R1$$

This will be done using two lemmas: *AddNilLeft* and *AddNilRight*. *AddNilFront* is shown below:

```
Lemma AddNilLeft:
```

$$(D1, D2 : (\mathtt{l}ist\ \ MLinProp))((\mathtt{E}mpty\ \hat{}\ \ D1 \vdash D2) \rightarrow (D1 \vdash D2)).$$

Each of these lemmas has a dual to eliminate the added Nils if necessary. Eliminating Nils will be done using *ElimNilLeft* and *ElimNilRight* lemmas. *ElimNilRight* is shown below.
`Lemma ElimNilRight:`

$$(D1, D2 : (\texttt{list}\texttt{MLinProp}))((D1 \vdash D2) \rightarrow (D1 \vdash \texttt{E}mpty \ \widehat{} \ D2)).$$

There are several other approaches to this problem. For example we could encode our sequents in such a way that left and right hand sides of the "$\vdash$" consist of only one list and no distinguished formula. Then the problem would be solved by making a singleton list out of the single formulas that appear on the RHS and LHS of "$\vdash$". While in this approach, list concatenation and singleton lists could be dealt with the same way as the encoding of [13].

5 Epistemic Applications

5.1 Proving a monomodal theorem

In this section we are going to prove a common property of most of the modal logics, i.e. *closure under material implication*:

$$K_1(A \multimap B) \vdash K_1A \multimap K_1B$$

To make the theorem more interesting, we slightly changed it to the following form:

$$K_1A, K_1(A \multimap B) \vdash K_1B$$

In informal terms, if an agent knows A and $A \multimap B$ then he also knows B. The proof tree is given below:

$$\dfrac{\dfrac{A \vdash A \quad B \vdash B}{A, (A \multimap B) \vdash B} \multimap L}{K_1A, K_1A \multimap B \vdash K_1B} \text{ TRule}$$

The proof in Coq is done in the same steps as the proof tree above. But we had to add *Nil* to the left of our sequents to be able to apply the *Implies Left* rule and the *Identity* axiom.The coq code is shown below:

```
Intros.
Apply TRule.
Apply AddNilLeft.
Apply ImpliesLeft.
Apply AddNilLeft.
Apply Identity.
Apply Identity.
```

Note that the above theorem says that agents know all the logical consequences of their knowledge . This makes the agents of a classical modal logic

idealized. But our modal logic by being substructural only allows agents to do proofs that do not exceed their cognitive capacities. It has embedded exponentials in structural rules to control the proofs so that they will not become unfeasible. This is one of the measures that make our logic a better candidate for an epistemic logic.

5.2 Proving a multimodal theorem

A standard puzzle of multi-modal logics is the The king, three wise men and 5 hats puzzle. It goes like this: a king has got three wise men and 5 hats: 2 green and 3 red. He asks the wise men to close their eyes and puts a hat on the head of each of them. Then asks them to open their eyes and poses a question to each of them in order. He asks the first man: 'Do you know the color of your hat?' He answers: 'No'. The same question is asked from the second man and he, too, answers No. But when the third man is asked the same question, he answers: ='Yes! The color of my hat is red'. How this is possible? We will show here that this conclusion can be made based on the information provided by the answers of previous wise men and using the sequent rules of our modal linear logic.In more formal terms we have: if agent three knows that agent one does not know the color of his hat, and he knows that agent two does not know the color of his hat and moreover he knows that agent two knows that agent one does not know the color of his hat, he will conclude the color of his own hat. Agent 3 ,therefore, knows the color of the hats of the other agents, the following three items that help him together with a good number of assumptions and some lemmas to conclude the color of his own hat, which is red:

1. Agent one does not know the color of his hat.

2. Agent two does not know the color of his hat.

3. Agent two knows that agent one does not know the color of his hat.

These information will help agent three to conclude that the color of his own hat is red. From (1) it can be concluded that at least one of the agents two and three wear a red hat. Because if both of them had green hats and since we only have two green hats, agent one would know the color of his hat. So a corollary of (1) is that agents 2 and three both know the following fact: At least one of agents two or three wears a red hat (or both of them do) This fact, together with (2) and (3) above help agent three to conclude that his hat is red. The fact that agent two does not know the color of his hat shows that agent three is not wearing a green hat. Because if this was the case, agent two, who knows that at least one of them is wearing a red hat, would have easily concluded the color of his own hat. In order to prove this theorem in the Coq, we have to define three agents, two color predicates and one definition.

1. Three agents:

```
agent1, agent2, agent3 :  nat.
```

2. Two color predicates:

   - `(red i):  the color of the hat of ith agent is red`
   - `(green i):  the color of the hat of ith agent is green`

3. Definition

   When each agent knows the color of his hat, it means he knows whether it is red or green. This can be shown using the additive Plus because it expresses a choice between two cases, where both of the cases cannot happen at the same time.

   (Lhat i): agent i knows that his hat is either red or green.

   or in Coq terms:

   `Definition Lhat := [`i`:  nat](`K_i `‘(red` i`)) ⊕ (`K_i `‘(green` i`)).`

We will use the same proof method as of Lescanne [9] with 6 axioms but in a linear logic environment:

1. AOne:Each hat is either red or green. This can again be shown using the additive Plus because (green i) and (red i) cannot both happen at the same time, i.e. each hat cannot be both red and green at the same time.

   `(`i`:nat)(Empty ⊢ ‘((green` i`)⊕ (red` i`))).`

2. ATwo:ATwo says that if two agents wear a green hat then the third one wears a red one. In this axiom, as opposed to the previous one, we want to be able to express that two cases happen at the same time, i.e. both agents wear a green hat. One of the Multiplicative connector seems a good option. We are going to use Par.

   `Axiom ATwo :  (‘((green agent2) ⅋ (green agent3)) ⊢ ‘(red agent1)).`

3. AThree: If agent2 has a green hat, then agent one knows it. The reason is obvious because he is seeing the hat of agent2.

   `(‘(green agent2) ⊢ ‘(agent1 K ‘(green agent2))).`

4. AFour: If agent3 has a green hat, then agent one knows it. The reason is obvious because he is seeing the hat of agent3.

   `(‘(green agent3) ⊢ ‘(agent1 K ‘(green agent3))).`

5. AFive : If an agent is wearing a red hat, then he is not wearing a green one.

   `(‘(red i) ⊢ ‘(Not (green i))).`

6. ASix : If an agent is wearing a green hat, then he is not wearing a red one.

```
(‘(green i)) ⊢ ‘(Not (red i))).
```

The theroem to be proved in sequent calculus is:

(agent2 K (Not (Lhat agent1))), (Not (Lhat agent2)) ⊢ (red agent3)

Or in Coq terms:

```
Theorem ThirdKnows :
(‘(Not (Lhat agent2)) ^ (‘(agent2 K ‘(Not(Lhat agent1)))) ⊢ ‘(red
agent3)).
```

The proof is done mostly with cuts. The proof tree and the Coq code is given in the appendix.

6 Discussion and Conclusion

The first version of this encoding was done in an intuitionistic fragment of modal linear logic. In that fragment sequents were limited to have only single formulas on their right hand side. Thus the first encoding of our logic had lists only in the left hand side of sequents and the right hand side contained only a single formula. This fragment did not have all the connectors of linear logic. In particular it missed Par ⅋ , the dual of Times ⊗. The reason being that sequent rules for Par, shown below are not intuitionistic:

$$Par \quad \frac{\Gamma \vdash A, B}{\Gamma \vdash A \parr B}\ R \qquad \frac{\Gamma_1, A \vdash C \quad \Gamma_2, B \vdash D}{\Gamma_1, \Gamma_2, A \parr B \vdash C, D}\ L$$

The problem with the intuitionistic fragment that dismissed Par was that in the proof of the puzzle at some stage we had to work with the dual of Times. Our choice of linear connectives in the encoding of the puzzle shown in the previous section, led us to prove the following sequent in the proccess of the proof of puzzle:

```
Not ((red 1) ⊗ (red 2)) ⊢ (green 1) ⅋ (green 2)
```

The proof seemed impossible in the fragment without ⅋ . In order to be able to solve the puzzle and to keep to our presented encoding, we decided to work with the full fragment of modal linear logic. Nevertheless, one way to stay in an intuitionistic fragment would be to re-phrase the puzzle and the axioms using the intuitionistic version pf ⅋ introduced in [2]. Changing to a classical fragment, we had to add lists to both sides of our sequents:

$$D1 \ ^\frown \ ‘A \vdash D2 \ ^\frown \ ‘B$$

As presented before, the puzzle was proven in this fragment. Witnessing this experiment, we believe that classical modal linear logic is a better candidate for such epistemic applications .

As conclusion, we have encoded a classical modal linear logic KDT_{lin} in higher order logic using proof assistant *Coq*. The encoding was done following the tow-level meta-reasoning in *Coq* presented in [6] and used before in a

previous encoding of intuitionistic linear logic in *Coq* [13]. The logic has been previously developed [10] by adding KD, T, and S4 modalities other than exponentials to Girard's classical linear logic. The encoding provided us with *Coq's* facilities to show how this logic can be successfully used in epistemic applications. However, the main result of this work was to show the feasibility of *Coq* as a proof assistant for the classical modal linear logic. The epistemic examples presented here demonstrated the benefits's of the encoding specially in dealing with lists of formulas on both sides of the sequent relation. A good suggestion for a further project would be to try to prove other puzzles of epistemic logic such as 'Muddy Children' or the puzzles that cannot be solved in classical epistemic logics and are only solvable in epistemic linear logic. Proving the latter puzzles will show the differences of linear logic over classical logics and thus will provide us with benifits of using linear logic as an epistemic logic over classical ones. One could also use *Coq* to compare the linear logic proofs with their classical logic counterparts. There are *Coq* encoding's for classical epistemic logics both in natural deduction [4] and Hilbert style systems [9]. Complexity analysis and proof automation using *Coq's* mechanisms are yet other further project that can be built on this work.

References

[1] A. Avron, 'Syntax and Semantics of Linear Logic', *Theoretical Computer Science* 57, pp.161-184, 1988.

[2] T.Brauner , and V. de Paiva , 'Cut-Elimination for Full Intuitionistic Linear Logic', `www.bricks.dk/RS/96/10`

[3] J. Dubucs, and M. Marion, 'Radical Antirealism and Substructural Logics', to appear in *Proceedings of LMPS99*, Dordrecht, Kluwer, 2002.

[4] P. de Wind, *Modal Logic in Coq*, M.Sc. Thesis, Faculty of Exact Sciences, Vrije Universiteit in Amesterdam, 2001

[5] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi, *Reasoning about Knowledge*, MIT Press, 1995

[6] A. Felty, 'Two-Level Meta Reasoning in Coq', *Fifteenth International Conference on Theorem Proving in Higher Order Logics*, Springer-Verlag LNCS 2410, 2002

[7] J-Y. Girard, 'Linear Logic', *Theoretical Computer Science*, 50, pp.1-102, 1987

[8] J. Hintikka, *Knowledge and Belief, An Introduction to the Logic of Two Notions*, N.Y., Cornell University Press, 1969

[9] P. Lescanne. ' Epistemic Logic in Higher Order Logic', `http://www.ens-lyon.fr/LIP/Pub/rr2001.html`, 2001.

[10] A. Martin. *Modal and Fixpoint Linear Logic*, M.Sc. Thesis, Department of Mathematics and Statistics, University of Ottawa, 2002.

[11] M. Ohnishi, and K. Matsumoto, 'Gentzen Method in Modal Calculi, I, *Osaka Mathematical Journal 9*, pp. 113-130, 1957

[12] M. Ohnishi, and K. Matsumoto, 'Gentzen Method in Modal Calculi, II, *Osaka Mathematical Journal 11*, pp. 115-120, 1959

[13] J. Powers, and C. Webster, 'Working with Linear Logic in Coq',12th International Conference on Theorem Proving in Higher Order Logics, 1999.

7 Appendix

7.1 Proof Tree

$$\dfrac{\dfrac{\Pi_1}{K_2(Not(Lhat1)) \vdash (red2) \otimes (red3)} \quad \dfrac{\dfrac{}{Not(Lhat2), (red2) \otimes (red3) \vdash (red2) \otimes (red3)}\ Identity \quad \dfrac{\dfrac{}{(red2), (red3) \vdash (red3)}\ Identity}{(red2) \otimes (red3) \vdash (red3)}\ \otimes L}{Not(Lhat2), (red2) \otimes (red3) \vdash (red3)}\ CUT}{K_2(Not(Lhat1)), Not(Lhat2) \vdash (red3)}\ CUT$$

Π_1 :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(red1) \vdash (red1)}\ Identity}{(red1) \vdash (red1) \oplus (green1)}\ \oplus R}{K_1(red1) \vdash K_1((red1) \oplus (green1))}\ T}{Not(Lhat1) \vdash Not(K_1(red1))}\ Unfold \quad \dfrac{\Pi_2}{Not(K_1(red1) \vdash (red2) \otimes (red3)}\ Negation}{Not(Lhat1) \vdash (red2) \otimes (red3)}\ CUT$$

$$\dfrac{}{K_2(Not(Lhat1)) \vdash (red2) \otimes (red3)}\ S4$$

Π_2 :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(green2) \vdash K_1(green2)}\ A3 \quad \dfrac{}{(green3) \vdash K_1(green3)}\ A4}{(green2) \parr (green3) \vdash K_1(green2), K_1(green3)}\ \parr L}{(green2) \parr (green3) \vdash K_1(green2) \parr K_1(green3)}\ \parr R \quad \dfrac{\dfrac{}{(green2) \parr (green3) \vdash (red1)}\ A2}{K_1(green2) \parr K_1(green3) \vdash K_1(red1)}\ T}{(green2) \parr (green3) \vdash K_1(red1)}\ CUT}{Not(K_1(red1)) \vdash Not((green2) \parr (green3))}\ Negation \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(red2) \vdash Not(green2)}\ A5}{(red2), (green2) \vdash Empty}\ Negation \quad \dfrac{\dfrac{}{(red3) \vdash Not(green3)}\ A5}{(red3), (green3) \vdash Empty}\ Negation}{(red2), (red3), (green2), (green3) \vdash Empty}\ CUT}{(red2), (red3) \vdash Not((green2) \parr (green3))}\ Negation}{(red2) \otimes (red3) \vdash Not((green2) \parr (green3))}\ \otimes L}{Not(K_1(red1)) \vdash (red2) \otimes (red3)}\ CUT$$

7.2 Coq Code

```
Section Hats.
Load MALL.
Variables red, green :  nat → MLinProp.
Variables agent1, agent2, agent3 :  nat.
Definition Lhat := [i:nat](i K ‘(red i)) ++ (i K ‘(green i)).
Axiom AOne :
(i:nat)(D : (list MLinProp))
( D ⊢ ‘ ((green i) %% (red i))).
Axiom ATwo :
(‘((green agent2) %% (green agent3)) ⊢ ‘(agent1 K ‘(red agent1))).
Axiom AThree :
(‘(green agent2) ⊢ ‘(agent1 K ‘(green agent2))).
Axiom AFour :
(‘(green agent3) ⊢ ‘(agent1 K ‘(green agent3))).
Axiom AFive :
(i :  nat)(‘(red i) ⊢ ‘(Not (green i))).
Axiom ASix :
(i :  nat)(‘(green i) ⊢ ‘(Not (red i))).
Lemma Duals :
(i , j :  nat)
(‘(Not ((green i)Proof.
Apply TimesLeft.
Apply AddNilRight.
Apply NegationRight.
Apply NegationRight.
Apply ParLeft.
Apply NegationRight.
Apply AFive.
Apply NegationRight.
Apply AFive.
Qed.
(* Main Theorem *)
Theorem ThirdKnows :
(‘(Not (Lhat agent2))  ^  (‘(agent2 K ‘(Not(Lhat agent1))))⊢ ‘(red agent3)).
(* Proof *)
Intros.
Apply Cut with (Times (red agent2) (red agent3)).
Apply AddNilLeft.
Apply S4Rule1.
Apply Cut with (Negation (agent1 K ‘(red agent1))).
Apply AddNilLeft.
Apply NegationLeft.
Apply AddNilRight.
Apply ExchangeRight.
```

```
Apply ElimNilRight.
Apply NegationRight.
Unfold Lhat.
Apply PlusRight1.
Apply Identity.
Apply Cut with (Negation (Par (green agent2) (green agent3))).
Apply AddNilLeft.
Apply NegationLeft.
Apply AddNilRight.
Apply ExchangeRight.
Apply ElimNilRight.
Apply NegationRight.
Apply ElimNilLeft.
Apply ATwo.
Apply ElimNilLeft.
Apply Duals.
Apply Cut with (red agent3).
Apply AddNilLeft.
Apply TimesLeft.
Apply ElimNilLeft.
Apply Identity.
Apply Identity.
End Hats.
```