Reasoning about Knowledge in Linear Logic: Modalities and Complexity

Mathieu Marion and Mehrnouche Sadrzadeh
Université du Québec à Montréal [*]

1 Introduction

In a recent paper, Jean-Yves Girard commented that "it has been a long time since philosophy has stopped intereacting with logic"[17]. Actually, it has not been such a "long" time since, e.g., Dag Prawitz and Michael Dummett developed philosophical arguments within the paradigm of Gentzen-style systems in favour of the adoption of intuitionistic logic. But recent developments within logic have left philosophers far behind. Prawitz's timely book *Natural Deduction* [**?**], along with a key result obtained at around the same time, the Curry-Howard isomorphism [21], initiated deep changes within logic; to wit, the development of the substructural logics [**?**, **?**]. Dummett developed within proof-theoretical semantics a mature version of his anti-realist challenge, closely allied to Prawitz's own philosophical considerations. But within the debate generated since by Dummett's anti-realism, philosophers have for the most part not paid much attention to the developments within logic since the 1970s. The almost complete absence of any discussion of linear logic within this context should convince any one that there is something seriously amiss here. The present paper is related to an attempt by Jacques Dubucs to provide a new impetus to the anti-realism debate by providing a radical anti-realist line of argument that also ties the debate more closely to issues of concern within substructural logics [**?**, 9, 10]. We shall only make brief remarks about his argument in section 1. Our intention is to push it a few steps further and explore the possibilities for a radically anti-realist epistemic logic.

In a nutshell, it has been suggested that a radical form of anti-realism should force one to look at structural rules within Gentzen-style systems that are responsible for the idealizations of the full structural logic. By 'idealization' we merely mean features of structural logic which allow for infinities to creep in, so to speak, and which should not go without notice within the interpretation of proofs as actions that has dominated much of the thinking about proof-theoretical semantics since the days of Prawitz and Dummett (e.g., in the work

[*]Département de philosophie, Université du Québec à Montréal, C.P. 8888, SUCC.A, Montréal, Québec, Canada H3C 3P8

of Girard or of Martin-Lof). Linear logic, by forbidding contraction and weakening as structural rules and by simultaneously introducing the exponentials ! and ? to recover them in some way, appears at first blush to be a promising candidate. Since its beginning, epistemic logic has been plagued with a serious case of idealization, the problem of logical omniscience. We shall propose here one new avenue for coping with this problem. Here again we build on ideas set forth by Dubucs [**?**]. This new approach requires that one develops a modal linear logic. In section 2, we shall discuss only two candidates for epistemic linear logic and eliminate one. The radical anti-realism advocated for here requires that issues concerning complexity should not be ignored, as they have been traditionally, precisely because, when discussing idealizations, complexity is part of the diagnosis. This is a serious problem for epistemic logics, where one seems not to be able to escape exponential complexity, especially in multi-agent systems. In the concluding section of our paper, we shall suggest one way out, namely the encoding of proofs in the proof assistant *Coq* [7].

The following considerations are of an exploratory and programmatic -in other words: philosophical- nature. There are no new results, except an coding in *Coq* of two different fragments of modal linear logic and the proof of the puzzle known as the *wise men puzzle* or *King, 3 wise men, and 5 hats puzzle*, which is a well-known version of the *muddy children puzzle*. This proof is given in the appendix. This paper is rather an informal discussion which should help to orient and to motivate future research. Although we shall go over some elementary points in order to make our philsoophical points, especially in the next section, some basic knowledge is presupposed.

2 Radical Anti-Realism and Linear Logic

The stance adopted here is that of *radical anti-realism*. It is the result of a radicalization of the anti-realist philosophy of Michael Dummett, on the basis of an argument already presented in other places [**?**, **?**, 10]. This argument certainly needs to be butressed by further philosophical considerations but this is not the place to do so. We should like merely to insist here on a few consequences from this argument, since they provide the initial impetus for the following. In these papers, it is argued that the traditional form of anti-realism propounded by Dummett is not satisfactory because, in a nutshell, it relies on the notion of assertability-conditions, where assertability is claimed to be effective *in principle*. This very notion is claimed to be as obscure as the realist notion of truth-conditions (which transcend our cognitive capacities) and it is argued that one should replace effectivity in principle by the notion of *feasibility* in practice. The argument is in essence as follows:

According to the definition of assertability-conditions, a statement is assertable if there exists an effective proof of it, that is a finite sequence of statements of which it is the last and of which every statement follow another as the result of an application of a rule of inference (there is only a finite number of such rules). Such as definition does not allow for an hypothetical being whose cognitive capacities would be such that it could, say, recognize the truth of a universal statement by inspection of an infinity of particular cases. The realist could still point out, however, that when the anti-realist admits of finite proofs that can be carried out merely in principle he does not fare much better than someone who admits of truth-conditions which transcend our cognitive capacities. Therefore, the definition must be such that our cognitive capacities must allow us always to recognize a sentence as assertable when it is, that is that one must be able to recognize the object $Pr(s)$ which is an effective proof of s, when there is one. This statement is ambiguous, since one may understand this either, as Dummett did, as the weaker claim that one has to be able recognize a proof of s when presented with one or as the stronger claim that one must be able to produce or reproduce the object $Pr(s)$. For the anti-realist really to distinguish his position from that of the realist on this rather crucial point, he must claim not only that circumstances in which an assertion is justified must be such that we should recognize them when we are in a position to do so, he must also claim that we must always be able in practice to put ourselves in such a position whenever such circumstances exist. Otherwise, it would be open for the realist to admit there should always exist circumstances under which we would recognize that an assertion is justified and merely to deny that we should always have the practical capacity to put ourselves in that position. To repeat, the weaker claim that one has to be able recognize a proof of s when presented with one won't do, because there may simply be situations where we could recognize a proof when presented with one, but we would never be able in practice to put ourselves in such a position. Therefore, in order to develop a coherent alternative to the realist, the anti-realist must develop a notion of assertability-conditions based on the fact that our own cognitive capacities must allow us not only always to recognize a sentence as assertable when it is, that is that one must be able to recognize the object $Pr(s)$ which is an effective proof of s, when there is one, but also to be able in practice to produce or construct the object $Pr(s)$. (In the jargon of computer scientists, once an answer to a problem is obtained, one may further produce a polynomial-time certificate while the algorithm had an exponential-time worst-case running time.)

There have already been arguments in favour of a radical form of constructivism known as 'strict finitism', e.g. [**?**], but Dubucs' proposal differs from these in two fundamental ways. First, the discussion is not anymore conducted in terms of the Hilbert-Style systems to which the notion of 'effectivity' is associated. It is conducted in terms of Gentzen-style systems, more precisely: it is about sequent calculi and their structural rules (and not even about introduction and elimination rules for natural deduction systems, as it was for Dummett and Prawitz). Secondly, Dubucs argues for feasibility in a more principled way, i.e., by looking at a weakening of structural rules as opposed to, say, a mere bounding of the length of computations. In other words, bounds should remain hidden, i.e., the logic for radical anti-realism should reflect limitations to human cognitive capacities in a 'structural' fashion. Furthermore, the key to the whole argument is seriously to take into account the *physical cost* of the proof, which is precisely the initial motivation for the development of linear logic, according to Girard (see, e.g., [16]).

The optional discharge in the case of the introduction rule for implication has the structural rule of weakening on the left as its counterpart. Relevant logic and linear logic both reject it, as opposed to intuitionistic logic, which merely distinguishes itself from the full structural logic by its restriction on weakening on the right. However, the counterpart of obligatory discharge is contraction on the left, which is still accepted by relevant logic but rejected by linear logic, where discharge is obligatory but not multiple. Now, once the focus is on strutural rules, a radical anti-realist may argue that there are no specific reasons to adopt intuitionistic logic, for which traditional anti-realists such as Dummett had argued. The relevance of relevant logic is not clear either, since the cause of idealizations, from the radical anti-realist point of view, is the rules of contraction, here shown with weakening:

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

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

But relevant logic consists in rejecting weakening (both left and right) while keeping contraction. Moreover, in absence of weakening, one needs to introduce *ad hoc* distributivity rules in order to keep to classical logic. In linear logic, rules for contraction do not disappear entirely (the resulting system would not be expressive enough); they reappear as the rules for special connectives, the exponentials. The sequent rules for exponentials are:

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

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

Linear *contraction* and *weakening* are shown below:

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

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

Classical *contraction* enables us to use a formula infinitely many times in a proof (this is an idealization). Classical *weakening*, on the other hand, allows us to bring unused hypotheses into our proofs, with that we may be left with unrelated hypotheses. In linear logic, the exponentials are used to control the structural rules of contraction and weakening. 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 $?A$. The idea behind this move in linear logic is to *control* the use of contraction, e.g., the length of proof search or of normalization procedures. This ability to control contraction should be a prime topic for investigation from our radical anti-realist standpoint.

We should wrap up this section with some basic remarks about other connectives, which will turn out useful in the following section. A sequent of the form $\Gamma \vdash \Delta$ in linear logic means that resources 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 (see [14] for a more detailed discussion). To overcome these ambiguities, linear logic introduces two distinct connectives for each of conjunction and disjunction, resp., the multiplicatives and the 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:

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

One should note that according to the follwoing sequent rules, the two multiplicatives are De Morgan duals of each other. 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 the following left and right sequent rules:

$$\text{Times} \quad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \otimes L \qquad \frac{\Gamma \vdash A, \Delta 1 \quad \Gamma \vdash B, \Delta 2}{\Gamma \vdash A \otimes B, \Delta 1, \Delta 2}\ \otimes R$$

$$\text{Par} \quad \frac{\Gamma 1, A \vdash C \quad \Gamma 2, A \vdash D}{\Gamma 1, \Gamma 2, A \parr B \vdash C, D}\ \parr L \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ \parr R$$

$$\text{Plus} \quad \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}\ \oplus L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ \oplus R1 \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ \oplus R2$$

$$\text{With} \quad \frac{\Gamma, A \vdash \Delta}{\Gamma, A \& B \vdash \Delta}\ \& L1 \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \& B \vdash \Delta}\ \& L2 \qquad \frac{\Gamma \vdash A, \Delta 1 \quad \Gamma \vdash B, \Delta 2}{\Gamma \vdash A \& B, \Delta 1, \Delta 2}\ \& R$$

$$\text{Implies} \quad \frac{\Gamma 1, B \vdash \Delta \quad \Gamma 2 \vdash A}{\Gamma 1, \Gamma 2, A \multimap B \vdash \Delta}\ \multimap L \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \multimap R$$

3 Epistemic Logic and Modal Linear Logic

Reasoning about knowledge is one of the many areas where problems about computational complexity cannot be eluded. The problem of logical omniscience in epistemic logic is a perfect case of an idealization in the above sense. It is usually presented in Hintikka's original Hilbert-style system [19]. We shall present a Gentzen-style version below. Informally the problem is this: 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: for example, one does not know all the consequences 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, if, for example, the computation involves exponential complexity. Here too, philosophers have not been able to engage with the issues raised by the logicians. Joseph Halpern's remark still stands today:

> ... reasoning about knowledge has found applications in such diverse fields as economics, linguistics, artificial intelligence, and computer science. While researchers in these areas have tended to look to philosophy for their initial inspiration, it has also been the case that their more practical concerns, which often centred around more computational issues such as the difficulty of computing knowledge, have not been treated in the philosophical literature.[**?**], p. 2

The literature contains many attempted solutions to the problem of logical omniscience, from Hintikka's own ideas about 'impossible possible worlds' [**?**], based on Rantala's 'urn' models [**?**, **?**], to the syntactical solutions of [**?**, **?**], Parikh's 'knowledge algorithms' [**?**, **?**], and the logic of 'awareness' [12]. This is not the place for a critical evaluation of these alternatives. It should be pointed out, however, that number of these solutions can be characterized by the wish to adhere come what may to the full structural logic: a sort of superstructure -one is tempted to say: epicycle- is then added to it within which one could talk about agents reasoning about knowledge without conceiving of them as omniscient. There are no reasons, except philosophical prejudice, not to explore the substructural world. We prefer here to follow Dubucs [9] and try and look for *a weaker logic relatively to which agents can be said to be omniscient, without this omniscience being problematic.* In that paper, Dubucs only discussed intuitionistic logic as a posssible alternative but in [**?**] linear logic is discussed. Intuitionistic logic is not an interesting candidate for us precisely because it does not keep contraction and weakening under control and, in that sense it is no more likely than classical logic (as the full structural logic) to be the weaker logic that we are looking for, relatively to which agents can be said to be omniscient. Further reasons for its inadequacy in the epistemic context will surface below.

The only attempt that we know of at developing an epistemic logic by using a substructural logic is by Hector Levesque [24, 25], who used relevant logic.

Levesque's approach is original in many respects. First, he distinguishes between 'implicit' and 'explicit' knowledge. According to Levesque, the possible worlds semantics is an idealization because it is not about what is known by an agent, but what is true given what is known. One must distinguish between what is known in this sense from what is *explicitely* known by the agent. Implicit knowledge is therefore defined as something that is true in all the worlds that agent considers as possible and explicit knowledge is what is known as true for the agent. Levesque introduces the operator I and E as, resp.,'$E_i\phi$ is true if ϕ is explicitely known' and '$I_i\phi$ is true if ϕ is implicitely in what is known'. (It is on the basis of this distinction that Fagin and Halpern introduced a logic of 'awareness' [**?**], where an agent knows explicitely ϕ if she is aware that ϕ and knows implicitely that ϕ.) Secondly, Levesque uses the situation semantics of [3] to deal with the explicit knowledge of agents. In a given situation, some formulas will have a truth-value assigned to them but it is possible some other formulas can have no truth-value. Levesque also uses the notion of an 'incoherent' situation, which is not compatible with any possible worlds. In such sitations some formulas can be seen as both true and false. Levesque identifies explicit knowledge as a set of situations and gives a semantics and a proof theory for it. The proof theory consists of propositional tautologies, modus ponens and axioms of propositional logic. He adds axioms for E and I which include closure under implicit knowledge. He also proves the following theorem:

$$\models (E_i\phi \supset E_i\psi) \quad \text{iff} \quad \phi \quad entails \quad \psi$$

The proof of this theorem can be found in [25]. This theorem allows Levesque to complete his axiomatization by using the axioms of entailment for explicit knowledge E.

From our radical anti-realist point of view, relevant logic is not an interesting alternative to the full structural logic. It may go further than intuitionistic logic in rejecting weakening on both right and left sides but it leaves contraction untouched and the resulting system are unappealing from a computational point of view. On top of this, Levesque's approach suffer from many defects of its own. To begin with, it is limited to only one agent and there seems to be no room for the multi-agent case, which is needed for a full epistemic logic. Secondly, a clear philosophical motivation for the distinction between 'implicit' and 'explicit' knowledge is lacking. Thirdly, the notion of an 'incoherent' situation needs to be clearly distinguished from that of impossible possible worlds. These last two defects would require a fuller philosophical discussion for which this is not the appropriate forum. It remains, however, that in absence of control over contraction within relevant logic, it is not clear that the idea that remain omniscient for their explicit knowledge really solves the problem of omniscience. For these reasons, it is worth looking at an *epistemic linear logic*. In order to deal with the logic of knowledge and belief, we must extend linear logic by introducing modalities. In this paper, we cannot do more that merely explore various ways of doing so. Semantical considerations cannot be truly dealt with at this stage. Jaakko Hintikka's seminal discussion in [19] is a model of clarity and elegance that has had no equivalent and we are certainly not yet in a position to present our own. But the following considerations should help us to select a candidate for further semantical and philosophical elaboration.

There are two main strategies for introducing modalities within linear logic. First, one could interpret exponentials as modalities, as in [2], the system would be a form of linear S4 with indexed modalities. However, the resulting connectives would both control contraction and weakening and serve as modalities. They would appear in both structural and modal rules; a very bizarre cocktail. Secondly, one could add modalities to fragments of linear logic. Some such combinations and their semantics have been studied in [27].In what follows, we shall assess only two of them, namely the multi-modal linear logic or MMLL of [22], which has been inspired by and shown to have applications in reasoning about location-dependent distributed network processes, and the system $KDT4_{Lin}$ developed in [27]. In what follows we first give a brief overview of both of these logics. (Some familiarity with both linear and modal logics is assumed.) We then give examples of the application of these logics to the problem of logical omniscience and the wise men puzzle. The proofs of these problems will be compared with each other and a short presentation of our encoding of modal linear logic in *Coq* will be given (the full proof tree and *Coq* code are in the Appendix). Finally, we shall give our reasons for choosing $KDT4_{lin}$, which is classical, over MMLL, which is intuitionistic.

MMLL uses the proof-search-as-computation paradigm where formulas of linear logic are seen as processes. The concurrency aspects and location of processes in the network are dealt by using the added modality L. The semantics is based on the resource-indexed model of [29], it is of a set of located resources with different combinations of these resources. These combinations correspond to the additive and multiplicative connectives of linear logic. An algebra of resources and a resource structure on this algebra are defined and the semantics is proved to be sound and complete in [22]. The BNF of the logic is:

$$A ::= a|!A|L_iA|A \otimes A|A \& A|A \multimap A|1$$

where:

- a is an atomic formula
- 1 is the unit for tensor $((A \otimes 1) = A)$
- L is the modal operator
- i is a natural number ranging over a denumerably infinite set
- L_iA intuitively means that resource A is available at location i.
- $\otimes$ and & are the multiplicative and additive connectives of linear logic.

(Note that this system does not contain negation or dual operators.) The sequent rules for this logic are the same as for linear logic except for the following two cases that are used for the congruences:

$$\frac{\Gamma, A' \vdash D \quad A \equiv A'}{\Gamma, A \vdash D} \equiv L \qquad\qquad \frac{\Gamma \vdash A \quad A \equiv A'}{\Gamma, \vdash A'} \equiv R$$

This system is intuitionistic, i.e., its sequents have a single formula on their right-hand side. There is no sequent rule for introduction and elimination of modalities. The only way to work with modalities is the following congruences over formulas:

1. $L_i(A \otimes B) \equiv L_iA \ \otimes \ L_iB$
2. $L_i(A \& B) \equiv L_iA \ \& \ L_iB$
3. $L_i(A \multimap B) \equiv L_iA \ \multimap \ L_iB$
4. $L_i!A \equiv !\mathrm{L}_iA$
5. $L_i1 \equiv 1$
6. $L_iL_jA \equiv L_jA$

The system MMLL has some advantages. First, it has the same contraction rules as linear logic and thus avoids idealizations. Secondly, as we can see from the syntax, this system has indexed modalities that make it a multi-modal system; one that can be applied to multi-agent systems. But the indexed modalities of this system are problematic, as opposed to those in $KDT4_{lin}$, which will be discussed below, because of the existence of the last congruence in the above list. This congruence allows for connections between the knowledge of different agents. In one direction it says that if agent1 knows that agent2 knows A, then agent2 himself knows A. In the other direction, it says that if agent1 one knows A, then there is another agent, say, agent2, who knows that agent1 knows A. This is a bit counterintuitive. One should note that it is not necessary that whenever I know A, there exists someone else that knows that I know A, but the existence of that other agent is not impossible. (In other words, there are no explicit quantifiers over the indexed modalities, so this congruence is not so counterintuitive.) Had it not been so counterintuitive, this congruence would have made MMLL of interest not only for reasoning about knowledge, but also for a discussion of the intersubjectivity in knowledge; it is a sort of iteration principle. However for reasons that will apprear below, MMLL is at any rate not suited for some crucial epistemic purposes.

$KDT4_{lin}$ has an algebraic semantics and it has been proven to be sound and complete in [27]. 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
- $1, \perp, \top$, 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_iA intuitively means that i knows A.

The sequent rules of this logic are the same as the full propositional classical linear logic, in which we take sequents $\Gamma \vdash \Delta$ for lists of formulas. These lists, together with the *Exchange* rule, will have the properties of multisets. The sequent calculus will also have three modal sequent rules shown below. These rules correspond to T, KD, and S4 axioms of the classical Hilbert-style modal logics:

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

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

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

Where Γ is a multiset of formulas and $K_i\Gamma$ is the multiset $\{K_iA | A \in \Gamma\}$. (Note that *Trule-Right* and *KDrule-Right* are the same, because we have chosen to work on one modality K_i as opposed to dual modalities K_i and B_i; the dismissed modality B_i would distinguish between the two rules.)

One should observe that all the connectives input linear propositions, but K_i takes as input a list of linear propositions. The modality sequent 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* where the modality K_i expresses the knowledge of agent i. For example K_1D intuitively means that *agent* 1 *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.

The system $KDT4_{lin}$ has many advantages. It avoids idealization, as it keeps the exponentials and also all the structural rules of linear logic. Hence, it is capable of controlling the resources by marking them with exponentials. It also fares better for epistemic purposes than Levesque's system in [24, 25] because it is *multi-modal*. There is also no trace of the problematic congruence that we found in MMLL.

We shall now encode both MMLL and $KDT4_{lin}$ in the proof assistant *Coq*, developped in [8] on the basis of the Calculus of Constructions (*CC*) of [7]. Systems can be encoded in *Coq*'s higher-order logic; these encodings allows us to state and prove theorems using facilities of this proof assistant. Intuitionistic linear logic has been previously encoded in *Coq*, in [32], by associating the constructs of *Coq* together with linear logic proofs. Modal logic, too, has been previously encoded in *Coq* in [23]. Our encoding method will be based on that of [12], in which the system to be encoded is treated as the *object logic* and *Coq*'s Calculus of Constructions (CC) as the *metalogic*.

We are here encoding for the first time modalities in classical linear logic. (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 two steps: (i) defining modal linear logic formulas and (ii) modal linear logic sequent rules inductively, using the set of inductive datatypes of *Coq*, and (iii) proving some lemmas to work with lists.

In the first phase, 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 in *Coq* is:

```
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.
```

Using *Coq's* syntax definition and pretty-printing facilities, we can give a notation to each of our modal linear connectives. This will allow us to infix and prefix our connectives.The *Coq* code for Bang, Times, and modality is given below. We are augmenting the grammar rules and giving pretty-printing rules to represent Bang as "!", Times as "*", and modality 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 in [8]).

```
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.

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

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 hypothesis Γ 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 function, which is either true or false, is defined as a *Coq* proposition *Prop*. The *Coq* code for *LinCons* is:

```
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. Following the encoding of [32], two symbols $\frown$ and ‘ are used to work with lists in *Coq*; $\frown$ is used to concatenate 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 $\rightarrow$ for deduction. For example the *Cut* rule:

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

is coded as:

```
| 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, consider the *Coq* code for Par Left and Times Right:

```
| ParLeft :
(A, B, C1 , C2 : MLinProp)(D1, D2 : (list MLinProp))
((D1 ⌢ ‘A ⊢ ‘ C1) → (D2 ⌢ ‘B ⊢ ‘C2) → (D1 ⌢ D2 ⌢
‘(A%%B) ⊢ ‘C1 ⌢ ‘C2))
|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 KD, T, and S4.The different thing about them is that the modal operator has two operands: an index i and a list of formulas D. $K_i D$ will be shown as iKD in Coq.For example the KD rule below:

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

will be code as:

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

In the third phase of our encoding we will deal with some lemmas to work with lists. Our sequent rules have lists to the right and left of the sequents. That will cause difficulty while working with the sequents that do not contain lists on one side or on both sides. For example sequents of the form $A \vdash A$ or $A \vdash A \oplus B$ are not accepted in our encoding. Moreover, a deduction using these sequents, which is an acceptable deduction, will not be accepted in our system.One such deduction would be:

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

To solve the problem we will have to make lists out of single formulas. This will be done by adding Nil lists to the left hand side of them. Applying these changes to the above deduction makes it look like:

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

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

$$\texttt{Lemma AddNilLeft} : (D1, D2 : (\texttt{list MLinProp}))$$

$$((\texttt{Empty} \mathbin{\hat{}} D1 \vdash D2) \rightarrow (D1 \vdash D2)).$$

Each of these lemmas has a dual to eliminate the added Nils. This is necessary because we are working with sequents with distinguished formulas. So we need to have a list and a single formula on both sides of the sequent.By adding Nil we will have sequents without distinguished formulas. So we have to eliminates the nils that we added before. Eliminating Nils will be done using *ElimNilLeft* and *ElimNilRight* lemmas. *ElimNilRight* is shown below.

$$\texttt{Lemma ElimNilRight} : (D1, D2 : (\texttt{list MLinProp}))$$

$$((D1 \vdash D2) \rightarrow (D1 \vdash \mathrm{E}mpty \mathbin{\hat{}} D2)).$$

List concatenation and singleton lists are dealt with the same way as the encoding of [32].

As examples of encoding, we shall state the problem of logical omniscience and the wise men puzzle as theorems and then prove them, in both MMLL and $KDT4_{lin}$. In [23], Lescanne has already encoded in *Coq* the latter along with the muddy children puzzle. But those are in Hilbert-style classical modal logic. The sequent calculus version of logical omniscience is the following:

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

The proof tree for the $KDT4_{lin}$is:

$$\dfrac{\dfrac{A \vdash A \quad B \vdash B}{A, (A \multimap B) \vdash B} \multimap L}{K_1 A, K_1 A \multimap B \vdash K_1 B} \text{KDRule}$$

The proof in *Coq* is done as in the above proof tree, using the sequent rules encoded in *Coq*. Some extra work has to be done while working with lists of formulas, namely adding *Nil* to the left and right of our sequents to be able to apply the *ImpliesLeft* and *ImpliesRight* rules and the *Identity* axiom. The *Coq* code is thus:

```
Intros.
Apply KDRule.
Apply AddNilLeft.
Apply ImpliesLeft.
Apply AddNilLeft.
Apply Identity.
Apply Identity.
```

The proof tree in MMLL is:

$$\dfrac{\dfrac{L_1A \vdash L_1A \quad L_1B \vdash L_1B}{L_1A, L_1A \multimap L_1B \vdash L_1B}\multimap L}{L_1A, L_1(A \multimap B) \vdash L_1B}\text{3d congruence}$$

One of the standard puzzles for multi-modal epistemic logic is 'wise men' or 'King, three wise men and 5 hats' puzzle (see [11], p. 12): *a king has 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 colour 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 colour of my hat is red'. How this is possible?* This conclusion is based on the information provided by the answers of previous wise men, together with the fact that each agent knows the color of the hats of the other agents except for himself. In more formal terms we have: if agent 3 knows that agent 1 does not know the colour of his hat, and he knows that agent 2 does not know the colour of his hat and moreover he knows that agent 2 knows that agent 1 does not know the colour of his hat, he will know the colour of his own hat. Therefore, agent 3 knows three things that help him, together with a good number of assumptions and some lemmas, to reach a conclusion about the colour of his own hat (red). These three things are:

1. Agent 1 does not know the color of his hat.
2. Agent 2 does not know the color of his hat.
3. Agent 2 knows that agent 1 does not know the color of his hat.

These three pieces of information will help agent 3 to conclude that the colour of his own hat is red. From (1) it can be concluded that at least one of the agents 2 and 3 wear a red hat. Indeed, if both of them had green hats, since we only have two green hats, agent 1 would know the colour of his hat. So a corollary of (1) is that agents 2 and 3 both know the following fact: At least one of agents 2 or 3 wears a red hat (or both of them do) This fact, together with (2) and (3) above help agent 3 to conclude that his hat is red. The fact that agent 2 does not know the colour of his hat shows that agent 3 is not wearing a green hat. Because if this were the case, agent 2, 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 *Coq*, we need three agents, two colour 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 $\oplus$ 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`))` $\oplus$ `(`K_i` ‘(green `i`)).`

We will use the proof method in [23], with linear logic axioms:

1. AOne:Each hat is either red or green. This can again be shown using the additive $\oplus$ 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)(`D` :  (list MLinProp))(`$D \vdash$` ‘((green `i`)`$\oplus$` (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. A multiplicative connector is called for and we are going to use $\parr$.

   `Axiom ATwo :  (‘((green agent2)` $\parr$ `(green agent3))` $\vdash$ `‘(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)` $\vdash$ `‘(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 theorem 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 are given in the Appendix.

An attempt at this encoding with MMLL will fail. This system is intuitionistic so the first encoding of our logic has lists only in the left-hand side of sequents and only single formulas on the right-hand side. Therefore, this fragment does not have all the connectives of linear logic. It misses ⅋ , the dual of ⊗, because the sequent rules for ⅋ are not intuitionistic:

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

The problem with this fragment is that in the proof of the puzzle we need at one stage the dual of ⊗. We had to prove the following sequent:

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

Thus, the puzzle cannot be proved in the fragment without ⅋ . This does not settle the matter entirely, as one could attempt to re-phrase the puzzle (without any loss of meaning) so that it could be expressed in MMLL and use the intuitionistic version of ⅋ introduced in [4], but this is mere speculation. Therefore, in order to be able to solve the puzzle, we had to work with the *full* modal linear logic. So we had to add lists to both sides of our sequents:

$$D1 \frown {}`A \vdash D2 \frown {}`B$$

One very important consequence of this is that there seem to be no real prospect for an intuitionistic epistemic linear logic.

4 Complexity

From our radical anti-realist point of view, computational complexity is to be taken seriously. Furthermore, the issue of complexity cannot be avoided when dealing with practical applications, e.g, in case of epistemic logic, applications in the domain of cognitive science or artificial intelligence. The key idea here is that the brain can be analysed as a computational system and one would propose models for cognitive activities in terms of computational tasks. But, to put it crudely, such tasks can hardly have an exponential lower bound, because one would not know how this task or cognitive activity is physically possible: idealizations must be avoided. Although the topic was hardly dealt with a mere 25 years ago, computational complexity is by now a well-studied phenomenon, with proof-theoretical measures of complexity, see, e.g., the survey in [33]. One obvious proposal here would be to limit oneself to polynomial time. (This has been suggested, e.g., in [26] for epistemic logic, because of applications to cognitive science.) There are candidates for this in linear logic, such as the Bounded Linear Logic (BLL) of [18], in which the use of exponentials is bounded in advance, or the more recent Light Linear Logic (LLL) of [17], that has a (locally) polynomial-time cut-elimination. Both BLL and LLL are strong enough to represent all polynomial-time functions. We would like to suggest, however, that this is not, *prima facie*, the right approach. We shall give here, in very brief outline, three arguments. These will hardly settle the question but we hope to initiate a discussion.

First, it seems that epistemic logic is a hopeless case from the point of view of complexity, especially if we deal with multi-agent systems. At any rate, MMLL and $KDT4_{lin}$ are mere variants on the full classical linear logic, which is exponential. Moreover, it is not clear if the system resulting from an hypothetical addition of modalities to BLL or LLL would remain polynomial-time. Secondly, there is a conflict, so to speak, between theory and practice. This can be illustrated by considering a well-known algorithm, the simplex method in linear programming. This is a perfectly constructive method: when optimal solutions to linear programming problems exist, it gives us an algorithm to compute them. However, this algorithm is exponential-time. Still, in practice it outperformed a polynomial-time algorithm and cases where the simplex run for an exponential amount of time hardly ever occurred in practice. Thirdly, on a more philosophical note, the issue of complexity is, from a proof-theoretical standpoint, rather paradoxical. Indeed, use of cut, which corresponds to the use of lemmas in ordinary mathematics, allows for proofs that can be taken in. Now, if we keep in mind applications to domains such as reasoning about knowledge, the proof-theoretical approach creates a paradoxical situation:

> The idea of lengths of proofs is quite amusing from the perspective of reasoning. It suggests that there are some statements that are true that we cannot understand in practice because it would take too long, and that there are statements which we can understand if we permit ourselves to use cuts and not otherwise[6], p.137

It is for these reasons, which need to be argued for in a more substantial manner (counterarguments easily spring to mind), that we chose to deal with the problem from a computational viewpoint and we chose to have an encoding in *Coq*. The computational approach to linear logic initiated in [1] links it to functional programming languages and the key here is an extension the Curry-Howard isomorphism [21], which establishes a correspondence between linear logic proofs and computer programs and allows us to see cut-elimination as computation. This is a powerful paradigm that has taken over proof theory but, although it has been argued for in, e.g., [28], it has hardly been noticed by philosophers. The Curry-Howard isomorphism is extended in [1] to classical as well as to the intuitionistic fragment of linear logic. Through this computational interpretation we are able to reduce the complexity of our proofs to the complexity of programs and we think that a first step here should be to limit programs to constuctive programs. In order to be able to get constructive programs out of proofs we need tools and one such tool is the proof assistant *Coq*, which is a higher-order logic based on the calculus of constructions, whose ancestors are to be found in de Bruijn's Automath, Girard's system F [13] and Martin-Löf's intuitionistic type theory [28]. The calculus of constructions enables us constructively to encode other logics in *Coq*. These logics are treated as object logics vs the metalogic, which is the higher-order logic of *Coq*. The key point here is that, once theorem-proving in these logics becomes automated in *Coq*, then one has constructive programmes [5]. Hence, the encoding of our two modal linear logics in *Coq* provides us with automated proofs which lead to constructive programs. Proof automation has not been examined in this paper but, as mentioned before, our encoding is similar to that of [32], where issues related to proof automation are discussed, with a context-handling system and a general proof strategy. Guidelines for context-handling are mentioned and used succesfully in [32]. The general proof strategy can be found in the linear logic programming approach in [20]. Once proofs are automated, *Coq* provides a mechanism for construction of programmes out of automated theorem proving. As we just pointed out, automated proofs in *Coq* are constructed proofs that are correct-by-construction. This also provides us with a decision algorithm. Although the issue about complexity is hardly settled, this suggests encoding in *Coq* as a step towards the right direction.

References

[1] S. Abramski, 'Computational Interpretations of Linear Logic', *Theoretical Computer Science*, vol. 111, 1993, pp.3-57

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

[3] J. Barwise, and J.Perry, *Situations and Attitudes*, Cambridge Mass., MIT Press, 1983

[4] T. Brauner, and V. de Paiva , 'Cut-Elimination for Full Intuitionistic Linear Logic', Technical Report 395, Computer Laboratory University of Cambridge and BRICKS, Denmark, May 1996

[5] J. Caldwell, 'Decidability Extracted: Synthesizing Correct-by-Construction Decision Prodecure from Constructive Proofs', Ph.D. thesis, Cornell University, 1998

[6] A. Carbone, and S. Semmes, 'Making Proofs without Modus Ponens: An Introduction to the Combinatorics and Complexity of Cut Elimination', *Bulletin of the American Mathematical Society*, vol. 34, 1997, pp.131-159.

[7] T. Coquand, and G. Huet, 'The Calculus of Constructions', *Information and Computation*, vol. 76, 1988, pp.95-120

[8] C. Cornes, J. Courant, J-C. Filliatre, G. Huet, P. Manoury, C. Murioz, C. Murthy, C. Parent, C. Paulin-Mohring, A. Saibi,and B. Werber, The Coq Proof Assistant Reference Manual,Version 7.4 , Rapport Technique 177, INRIA, 1999-2003

[9] J. Dubucs, 'Feasibility in Logic', *Synthese*, vol. 132, 2002, pp.213-237

[10] J. Dubucs, and M. Marion, 'Radical Antirealism and Substructural Logics', to appear in *Proceedings of the contributed papers of LMPS99*, Dordrecht, Kluwer, 2003

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

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

[13] J.-Y. Girard, 'Une extension de l'interprétation de Gödel a l'analyse et son application a l'elimination des coupures dans l'analyse et la théorie des types', in J.-E. Fenstad (ed.) *Proceedings of the Second Scandinavian Logic Symposium*, Amsterdam, North-Holland, 1970, pp.63-92

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

[15] J-Y. Girard, 'Light Linear Logic', in D.Leivant(ed.), *Logic and Computational Complexity*, Berlin, Springer, 1995, pp.145-176

[16] J-Y. Girard, 'Linear Logic: Its Syntax and Semantics', in J-Y. Girard, Y.Lafont and L.Regnier (eds.), *Advances in Linear Logic*, Cambridge, Cambridge University Press,1995, pp.1-42

[17] J-Y. Girard, 'On the Meaning of Logical Rules I:Syntax vs. Semantics', `http://iml.univ-mrs.fr/ girard/Articles.html`, 1998

[18] J-Y. Girard, A.Scedrov and P.Scott, 'Bounded Linear Logic, A Modular Approach to Polynomial-Time Computability', *Theoretical Computer Science*, vol.97, 1992, pp.1-66

[19] J. Hintikka, *Knowledge and Belief, An Introduction to the Logic of Two Notions*, N.Y., Cornell University Press, 1962

[20] J.S. Hodas, 'Lolli: An Extension of Lambda Prolog with Linear Logic Context Management', *Workshop on the lambda Prolog Programming Language*, Philadelphia, 1992, pp.159-168

[21] W. A. Howard, 'The Formulae-as-Types Notion of Construction', in J.P.Seldin and J.R. Hindley, (eds.), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism* , Academic Press, London,1980, pp.479-490

[22] N. Kobayashi, T. Shimizu, and A. Yonezawa, 'Distributed Concurrent Linear Logic Programming', *Theoretical Computer Science*, vol. 227, 1999, pp.185-220

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

[24] H. Levesque, 'A Logic of Implicit and Explicit Belief', *Proceedings of the National Conference on Artificial Intelligence*, AAAI-84, pp.192-202, 1984

[25] H. Levesque, 'A Logic of Implicit and Explicit Belief', Fairchild Laboratory for Artificial Intelligence Research, Technical Report, 1984

[26] H.Levesque, 'Logic and the Complexity of Reasoning', *Journal of Philosophical Logic*, vol. 17, 1988, pp. 355-389.

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

[28] P. Martin-Löf, *Intuitionistic Type Theory*, Naples, Bibliopolis, 1984.

[29] R. Milner, J.Parrow, and D.Walker, 'A Calculus of Mobile Processes I,II', *Information and Computation*, vol.100, 1992, pp.1-77

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

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

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

[33] A. Urquhart, 'The Complexity of Propositional Proofs', *Bulletin of Symbolic Logic*, vol. 1, 1995, pp. 425-467.

5 Appendix: Proving the King, three wisement, and five hats puzzle

5.1 Proof Tree

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

Π_1 :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Identity}{(red1) \vdash (red1)}\,\text{Identity}}{(red1) \vdash (red1) \oplus (green1)}\,\oplus\text{R}}{K_1(red1) \vdash K_1((red1) \oplus (green1))}\,\text{KD}}{K_1(red1) \vdash (Lhat1)}\,\text{Unfold}}{Not(Lhat1) \vdash Not(K_1(red1)}\,\text{Negation} \quad \dfrac{\Pi_2}{Not(K_1(red1) \vdash (red2) \otimes (red3)}}{Not(Lhat1) \vdash (red2) \otimes (red3)}\,\text{CUT}}{K_2(Not(Lhat1)) \vdash (red2) \otimes (red3)}\,\text{S4}$$

Π_2 :

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

5.2 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.
```