Metareasoning for multi-agent epistemic logics

Konstantine Arkoudas and Selmer Bringsjord

RPI
{arkouk,brings}@rpi.edu

Abstract. We present an encoding of a sequent calculus for a multi-agent epistemic logic in Athena, an interactive theorem proving system for many-sorted first-order logic. We then use Athena as a metalanguage in order to reason about the multi-agent logic an as object language. This facilitates theorem proving in the multi-agent logic in several ways. First, it lets us marshal the highly efficient theorem provers for classical first-order logic that are integrated with Athena for the purpose of doing proofs in the multi-agent logic. Second, unlike model-theoretic embeddings of modal logics into classical first-order logic, our proofs are directly convertible into native epistemic logic proofs. Third, because we are able to quantify over propositions and agents, we get much of the generality and power of higher-order logic even though we are in a first-order setting. Finally, we are able to use Athena's versatile tactics for proof automation in the multi-agent logic. We illustrate by developing a tactic for solving the generalized version of the wise men problem.

1 Introduction

Multi-agent modal logics are widely used in Computer Science and AI. Multi-agent epistemic logics, in particular, have found applications in fields ranging from AI domains such as robotics, planning, and motivation analysis in natural language [14]; to negotiation and game theory in economics; to distributed systems analysis and protocol authentication in computer security [17, 32]. The reason is simple—intelligent agents must be able to reason about knowledge. It is therefore important to have efficient means for performing machine reasoning in such logics. While the validity problem for most propositional modal logics is of intractable theoretical complexity[1], several approaches have been investigated in recent years that have resulted in systems that appear to work well in practice. These approaches include tableau-based provers, SAT-based algorithms, and translations to first-order logic coupled with the use of resolution-based automated theorem provers (ATPs). Some representative systems are FaCT [25], KsatC [15], TA [26], LWB [24], and MSPASS [38].

Translation-based approaches (such as that of MSPASS) have the advantage of leveraging the tremendous implementation progress that has occurred over

[1] For instance, the validity problem for multi-agent propositional epistemic logic is PSPACE-complete [19]; adding a common knowledge operator makes the problem EXPTIME-complete [22].

the last decade in first-order theorem proving. Soundness and completeness are ensured by the soundness and completeness of the resolution prover (once the soundness and completeness of the translation have been shown), while a decision procedure is automatically obtained for any modal logic that can be translated into a decidable fragment of first-order logic (such as the two-variable fragment). Furthermore, the task of translating from a modal logic to the classical first-order setting is fairly straightforward (assuming, of course, that the class of Kripke frames captured by the modal logic is first-order definable [18]; modal logics such as the Gödel-Löb logic of provability in first-order Peano arithmetic would require translation into second-order classical logic). For instance, the well-known formula $[\Box P \wedge \Box(P \Rightarrow Q)] \Rightarrow \Box Q$ becomes

$$\begin{aligned}\forall\, w_1 \,.\, [&(\forall\, w_2 \,.\, R(w_1,w_2) \Rightarrow P(w_2)) \wedge \\ &(\forall\, w_2 \,.\, R(w_1,w_2) \Rightarrow P(w_2) \Rightarrow Q(w_2))] \Rightarrow (\forall\, w_2 \,.\, R(w_1,w_2) \Rightarrow Q(w_2))\end{aligned}$$

Here the variables w_1 and w_2 range over possible worlds, and the relation R represents Kripke's accessibility relation. A constant propositional atom P in the modal language becomes a unary predicate $P(w)$ that holds (or not) for a given world w.

This is the (naive) classical translation of modal logic into first-order logic [19], and we might say that it is a *semantic* embedding, since the Kripke semantics of the modal language are explicitly encoded in the translated result. This is, for instance, the approach taken by McCarthy in his "Formalizing two puzzles involving knowledge" [31]. A drawback of this approach is that proofs produced in the translated setting are difficult to convert back into a form that makes sense for the user in the original modal setting (altough alternative translation techniques such as the functional translation to path logic can rectify this in some cases [39]). Another drawback is that if a result is not obtained within a reasonable amount of time—which is almost certain to happen quite often when no decision procedure is available, as in first-order modal logics—then a batch-oriented ATP is of little help to the user due to its "low bandwidth of interaction" [13].

In this paper we explore another approach: We embed a multi-agent epistemic logic into many-sorted first-order logic in a proof-theoretic rather than in a model-theoretic way. [2] Specifically, we use the interactive theorem proving system Athena [2]—briefly reviewed in the Appendix—to encode the formulas of the epistemic logic along with the inference rules of a sequent calculus for it. Hence first-order logic becomes our metalanguage and the epistemic logic becomes our object language. We then use standard first-order logic (our metalanguage) to reason about proofs in the object logic. In effect, we end up reasoning about reasoning—hence the term *metareasoning*. Since our metareasoning occurs at the standard first-order level, we are free to leverage existing theorem-proving systems for automated deduction. In particular, we make heavy use of Vampire

[2] This paper treats a propositional logic of knowledge, but the technique can be readily applied to full first-order multi-agent epistemic logic, and indeed to hybrid multi-modal logics, e.g., combination logics for temporal and epistemic reasoning.

[41] and Spass [42], two cutting-edge resolution-based ATPs that are seamlessly integrated with Athena.

Our approach has two additional advantages. First, it is trivial to translate the constructed proofs into modal form, since the Athena proofs are already *about proofs in the modal logic*. Second, because the abstract syntax of the epistemic logic is explicitly encoded in Athena, we can quantify over propositions, sequents, and agents. Accordingly, we get the generalization benefits of higher-order logic even in a first-order setting. This can result in significant efficiency improvements. For instance, in solving the generalized wise men puzzle it is necessary at some point to derive the conclusion $M_2 \vee \cdots \vee M_n$ from the three premises $\neg K_\alpha(M_1)$, $K_\alpha(\neg(M_2 \vee \cdots \vee M_n) \Rightarrow M_1)$, and

$$\neg(M_2 \vee \cdots \vee M_n) \Rightarrow K_\alpha(\neg(M_2 \vee \cdots \vee M_n))$$

where $M_1, \ldots, M_n$ are atomic propositions and α is an epistemic agent, $n > 1$. In the absence of an explicit embedding of the epistemic logic, this would have to be done with a tactic that accepted a list of propositions $[M_1 \cdots M_n]$ as input and performed the appropriate deduction dynamically, which would require an amount of effort quadratic in the length of the list. By contrast, in our approach we are able to formulate and prove a "higher-order" lemma stating

$$\forall\ P, Q, \alpha\ .\ \{\neg K_\alpha(P), K_\alpha(\neg Q \Rightarrow P), \neg Q \Rightarrow K_\alpha(\neg Q)\} \vdash Q$$

Obtaining the desired conclusion for any given $M_1, \ldots, M_n$ then becomes a matter of instantiating this lemma with $P \mapsto M_1$ and $Q \mapsto M_2 \vee \cdots \vee M_n$. We have thus reduced the asymptotic complexity of our task from quadratic time to constant time.

But perhaps the most distinguishing aspect of our work is our emphasis on *tactics*. Tactics are proof algorithms, which, unlike conventional algorithms, are guaranteed to produce sound results. That is, if and when a tactic outputs a result P that it claims to be a theorem, we can be assured that P is indeed a theorem. Tactics are widely used for proof automation in first- and higher-order proof systems such as HOL [21] and Isabelle [35]. In Athena tactics are called *methods*, and are particularly easy to formulate owing to Athena's Fitch-style natural deduction system and its assumption-base semantics [3]. A major goal of our research is to find out how easy—or difficult—it may be to automate multi-agent modal logic proofs with tactics. Our aim is not to obtain a completely automatic decision procedure for a certain logic (or class of logics), but rather to enable efficient interactive—i.e., semi-automatic—theorem proving in such logics for challenging problems that are beyond the scope of completely automatic provers. In this paper we formulate an Athena method for solving the generalized version of the wise men problem (for any given number of wise men). The relative ease with which this method was formulated is encouraging.

The remainder of this paper is structured as follows. In the next section we present a sequent calculus for the epistemic logic that we will be encoding. In Section 3 we present the wise men puzzle and formulate an algorithm for solving the generalized version of it in the sequent calculus of Section 2. In Section 4

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\ [\wedge\text{-}I] \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P}\ [\wedge\text{-}E_1] \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}\ [\wedge\text{-}E_2]$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\ [\vee\text{-}I_1] \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\ [\vee\text{-}I_2]$$

$$\frac{\Gamma \vdash P_1 \vee P_2 \quad \Gamma, P_1 \vdash Q \quad \Gamma, P_2 \vdash Q}{\Gamma \vdash Q}\ [\vee\text{-}E]$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q}\ [\Rightarrow\text{-}I] \qquad \frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}\ [\Rightarrow\text{-}E]$$

$$\frac{\Gamma \vdash \neg\neg P}{\Gamma \vdash P}\ [\neg\text{-}E] \qquad \frac{\Gamma, P \vdash \bot}{\Gamma \vdash \neg P}\ [\neg\text{-}I] \qquad \frac{}{\Gamma, P \vdash P}\ [\mathit{Reflex}]$$

$$\frac{\Gamma \vdash P}{\Gamma \cup \Gamma' \vdash P}\ [\mathit{Dilution}] \qquad \frac{\Gamma \vdash P \wedge \neg P}{\Gamma \vdash \bot}\ [\bot\text{-}I] \qquad \frac{}{\Gamma \vdash \top}\ [\top\text{-}I]$$

Fig. 1. Inference rules for the propositional connectives.

we discuss the Athena encoding of the epistemic logic and present the Athena method for solving the generalized wise men problem. Finally, in Section 5 we consider related work.

2 A sequent formulation of a multi-agent epistemic logic

We will use the letters P, Q, $R, \ldots$, to designate arbitrary *propositions*, built according to the following abstract grammar:

$$P ::= A \mid \top \mid \bot \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \mid K_\alpha(P) \mid C(P)$$

where A and α range over a countable set of atomic propositions ("atoms") and a primitive domain of *agents*, respectively. Propositions of the form $K_\alpha(P)$ and $C(P)$ are read as follows:

$K_\alpha(P)$: *agent α knows proposition P*

$C(P)$: *it is common knowledge that P holds*

By a *context* we will mean a finite set of propositions. We will use the letter Γ to denote contexts. We define a *sequent* as an ordered pair $\langle \Gamma, P \rangle$ consisting of a context Γ and a proposition P. A more suggestive notation for such a sequent

$$\frac{}{\Gamma \vdash [K_\alpha(P \Rightarrow Q)] \Rightarrow [K_\alpha(P) \Rightarrow K_\alpha(Q)]}\ [K] \qquad \frac{}{\Gamma \vdash K_\alpha(P) \Rightarrow P}\ [\mathrm{T}]$$

$$\frac{\emptyset \vdash P}{\Gamma \vdash C(P)}\ [C\text{-}I] \qquad \frac{}{\Gamma \vdash C(P) \Rightarrow K_\alpha(P)}\ [C\text{-}E]$$

$$\frac{}{\Gamma \vdash [C(P \Rightarrow Q)] \Rightarrow [C(P) \Rightarrow C(Q)]}\ [C_K] \qquad \frac{}{\Gamma \vdash C(P) \Rightarrow C(K_\alpha(P))}\ [R]$$

Fig. 2. Inference rules for the epistemic operators.

is $\Gamma \vdash P$. Intuitively, this is a judgment stating that P follows from Γ. We will write P, Γ (or Γ, P) as an abbreviation for $\Gamma \cup \{P\}$. The sequent calculus that we will use consists of a collection of inference rules for deriving judgments of the form $\Gamma \vdash P$. Figure 1 shows the inference rules that deal with the standard propositional connectives. This part is standard (e.g., it is very similar to the sequent calculus of Ebbinghaus et al. [16]). In addition, we have some rules pertaining to K_α and C, shown in Figure 2.

Rule $[K]$ is the sequent formulation of the well-known *Kripke axiom* stating that the knowledge operator distributes over conditionals. Rule $[C_K]$ is the corresponding principle for the common knowledge operator. Rule $[T]$ is the "truth axiom": an agent cannot know false propositions. Rule $[C_I]$ is an introduction rule for common knowledge: if a proposition P follows from the empty set of hypotheses, i.e., if it is a tautology, then it is commonly known. This is the common-knowledge version of the "omniscience axiom" for single-agent knowledge which says that $\Gamma \vdash K_\alpha(P)$ can be derived from $\emptyset \vdash P$. We do not need to postulate that axiom in our formulation, since it follows from $[C\text{-}I]$ and $[C\text{-}E]$. The latter says that if it is common knowledge that P then any (every) agent knows P, while $[R]$ says that if it is common knowledge that P then it is common knowledge that (any) agent α knows it. $[R]$ is a reiteration rule that allows us to capture the recursive behavior of C, which is usually expressed via the so-called "induction axiom"

$$C(P \Rightarrow E(P)) \Rightarrow [P \Rightarrow C(P)]$$

where E is the shared-knowledge operator. Since we do not need E for our purposes, we omit its formalization and "unfold" C via rule $[R]$ instead.
We state a few lemmas that will come handy later:

Lemma 1 (Cut). *If $\Gamma_1 \vdash P_1$ and $\Gamma_2, P_1 \vdash P_2$ then $\Gamma_1 \cup \Gamma_2 \vdash P_2$.*

Proof: Assume $\Gamma_1 \vdash P_1$ and $\Gamma_2, P_1 \vdash P_2$. Then, by $[\Rightarrow\text{-}I]$, we get $\Gamma_2 \vdash P_1 \Rightarrow P_2$. Further, by dilution, we have $\Gamma_1 \cup \Gamma_2 \vdash P_1 \Rightarrow P_2$ and $\Gamma_1 \cup \Gamma_2 \vdash P_1$. Hence, by $[\Rightarrow\text{-}E]$, we obtain $\Gamma_1 \cup \Gamma_2 \vdash P_2$. □

The proofs of the remaining lemmas are equally simple exercises:

Lemma 2 ($\Rightarrow$-transitivity). *If $\Gamma \vdash P_1 \Rightarrow P_2$, $\Gamma \vdash P_2 \Rightarrow P_3$ then $\Gamma \vdash P_1 \Rightarrow P_3$.*

Lemma 3 (contrapositive). *If $\Gamma \vdash P \Rightarrow Q$ then $\Gamma \vdash \neg Q \Rightarrow \neg P$.*

Lemma 4. *(a) $\emptyset \vdash (P_1 \vee P_2) \Rightarrow (\neg P_2 \Rightarrow P_1)$; and* (b) *$\Gamma \vdash C(P_2)$ whenever $\emptyset \vdash P_1 \Rightarrow P_2$ and $\Gamma \vdash C(P_1)$.*

Lemma 5. *For all P, Q, and Γ, $\Gamma \vdash [C(P) \wedge C(Q)] \Rightarrow C(P \wedge Q)$.*

3 The generalized wise men puzzle

Consider first the three-men version of the puzzle:

> Three wise men are told by their king that at least one of them has a white spot on his forehead. In reality, all three have white spots on their foreheads. We assume that each wise man can see the others' foreheads but not his own, and thus each knows whether the others have white spots. Suppose we are told that the first wise man says, "I do not know whether I have a white spot," and that the second wise man then says, "I also do not know whether I have a white spot." Now consider the following question: Does the third wise man now know whether or not he has a white spot? If so, what does he know, that he has one or doesn't have one?

This version is essentially identical to the muddy-children puzzle, the only difference being that the declarations of the wise men are made sequentially, whereas in the muddy-children puzzle the children proclaim what they know (or not know) in parallel at every round.

In the generalized version of the puzzle we have an arbitrary number $n+1$ of wise men $w_1, \ldots, w_{n+1}$, $n \geq 1$. They are told by their king that at least one them has a white spot on his forehead. Again, in actuality they all do. And they can all see one another's foreheads, but not their own. Supposing that each of the first n wise men, $w_1, \ldots, w_n$, sequentially announces that he does not know whether or not he has a white spot on his forehead, the question is what would the last wise man w_{n+1} report.

For all $n \geq 1$, it turns out that the last—$(n+1)^{st}$—wise man knows he is marked. The case of two wise men is simple. The reasoning runs essentially by contradiction. The second wise man reasons as follows:

> Suppose I were not marked. Then w_1 would have seen this, and knowing that at least one of us is marked, he would have inferred that he was the marked one. But w_1 has expressed ignorance; therefore, I must be marked.

Consider now the case of $n = 3$ wise men w_1, w_2, w_3. After w_1 announces that he does not know that he is marked, w_2 and w_3 both infer that at least one of them is marked. For if neither w_2 nor w_3 were marked, w_1 would have seen this and would have concluded—and stated—that he was the marked one, since he knows that at least one of the three is marked. At this point the puzzle reduces to the two-men case: both w_2 and w_3 know that at least one of them is marked, and then w_2 reports that he does not know whether he is marked. Hence w_3 proceeds to reason as previously that he is marked.

In general, consider $n+1$ wise men $w_1, \ldots, w_n, w_{n+1}, n \geq 1$. After the first j wise men $w_1, \ldots, w_j$ have announced that they do not know whether they are marked, for $j = 1, \ldots, n$, the remaining wise men $w_{j+1}, \ldots, w_{n+1}$ infer that at least one of them is marked. This holds for $j = n$ as well, which means that the last wise man w_{n+1} will infer (and announce, owing to his honesty) that he is marked.

The question is how to formalize this in our logic. Again consider the case of two wise men w_1 and w_2. Let $M_i, i \in \{1, 2\}$ denote the proposition that w_i is marked. For any proposition P, we will write $K_i(P)$ as an abbreviation for $K_{w_i}(P)$. We will only need three premises:

$$\begin{aligned}
S_1 &= C(\neg K_1(M_1)) \\
S_2 &= C(M_1 \vee M_2) \\
S_3 &= C(\neg M_2 \Rightarrow K_1(\neg M_2))
\end{aligned}$$

The first premise says that it is common knowledge that the first wise man does not know whether he is marked. Although it sounds innocuous, note that a couple of assumptions are necessary to obtain this premise from the mere fact that w_1 has announced his ignorance. First, truthfulness—we must assume that the wise men do not lie, and further, that each one of them knows that they are all truthful. And second, each wise man must know that the other wise men will hear the announcement and believe it. Premise S_2 says that it is common knowledge that at least one of the wise men is marked. Observe that the announcement by the king is crucial for this premise to be justified. The two wise men can see each other and thus they individually know $M_1 \vee M_2$. However, each of them may not know that the other wise man knows that at least one of them is marked. For instance, w_1 may believe that he is not marked, and even though he sees that w_2 is marked, he may believe that w_2 does not know that at least one of them is marked, as w_2 cannot see himself. Finally, premise S_3 states that it is common knowledge that if w_2 is *not* marked, then w_1 will know it (because w_1 can see w_2). From these three premises we are to derive the conclusion $C(M_2)$—that it is common knowledge that w_2 is marked. Symbolically, we need to derive the judgment $\{S_1, S_2, S_3\} \vdash C(M_2)$. If we have encoded the epistemic propositional logic in a predicate calculus, then we can achieve this immediately by instantiating Lemma 7 below with $\alpha \mapsto w_1$, $P \mapsto M_1$ and $Q \mapsto M_2$—without performing any inference whatsoever. This is what we have done in Athena.

For the case of $n = 3$ wise men our set of premises will be:

$$\begin{aligned}
S_1 &= C(\neg K_1(M_1))\\
S_2 &= C(M_1 \vee M_2 \vee M_3)\\
S_3 &= C(\neg(M_2 \vee M_3) \Rightarrow K_1(\neg(M_2 \vee M_3)))\\
S_4 &= C(\neg K_2(M_2))\\
S_5 &= C(\neg M_3 \Rightarrow K_2(\neg M_3))
\end{aligned}$$

Consider now the general case of $n + 1$ wise men $w_1, \ldots, w_n, w_{n+1}$. For any $i = 1, \ldots, n$, define

$$\begin{aligned}
S_1^i &= C(\neg K_i(M_i))\\
S_2^i &= C(M_i \vee \cdots \vee M_{n+1})\\
S_3^i &= C(\neg(M_{i+1} \vee \cdots \vee M_{n+1}) \Rightarrow K_i(\neg(M_{i+1} \vee \cdots \vee M_{n+1})))
\end{aligned}$$

and $S_2^{n+1} = C(M_{n+1})$. The set of premises, Ω_{n+1}, can now be defined as

$$\Omega_{n+1} = \{C(M_1 \vee \cdots \vee M_{n+1})\} \bigcup_{i=1}^{n} \{S_1^i, S_3^i\}$$

Hence Ω_{n+1} has a total of $2n + 1$ elements. Note that S_2^1 is the commonly known disjunction $M_1 \vee \cdots \vee M_{n+1}$ and a known premise, i.e., a member of Ω_{n+1}. However, S_2^i for $i > 1$ is *not* a premise. Rather, it becomes derivable after the i^{th} wise man has made his announcement. Managing the derivation of these propositions and eliminating them via applications of the cut is the central function of the algorithm below. Before we present the algorithm we state a couple of key lemmas.

Lemma 6. *Consider any agent α and propositions P, Q, and let R_1, R_2, R_3 be the following three propositions:*

1. $R_1 = \neg K_\alpha(P)$;
2. $R_2 = K_\alpha(\neg Q \Rightarrow P)$;
3. $R_3 = \neg Q \Rightarrow K_\alpha(\neg Q)$

Then $\{R_1 \wedge R_2 \wedge R_3\} \vdash Q$.

Proof. By the following sequent derivation:

1.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_1$	[*Reflex*], $\wedge$-E_1
2.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_2$	[*Reflex*], $\wedge$-E_1, $\wedge$-E_2
3.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_3$	[*Reflex*], $\wedge$-E_2
4.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash K_\alpha(\neg Q) \Rightarrow K_\alpha(P)$	2, [K], $\Rightarrow$-E
5.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg Q \Rightarrow K_\alpha(P)$	3, 4, Lemma 2
6.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg K_\alpha(P) \Rightarrow \neg\neg Q$	5, Lemma 3
7.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg\neg Q$	6, 1, $\Rightarrow$-E
8.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash Q$	7, [$\neg$-E]

$\square$

Lemma 7. *Consider any agent α and propositions P, Q. Define R_1 and R_3 as in Lemma 6, let $R_2 = P \vee Q$, and let $S_i = C(R_i)$ for $i = 1, 2, 3$. Then $\{S_1, S_2, S_3\} \vdash C(Q)$.*

Proof. Let $R_2' = \neg Q \Rightarrow P$ and consider the following derivation:

1.	$\{S_1, S_2, S_3\} \vdash S_1$	[*Reflex*]
2.	$\{S_1, S_2, S_3\} \vdash S_2$	[*Reflex*]
3.	$\{S_1, S_2, S_3\} \vdash S_3$	[*Reflex*]
4.	$\emptyset \vdash (P \vee Q) \Rightarrow (\neg Q \Rightarrow P)$	Lemma 4a
5.	$\{S_1, S_2, S_3\} \vdash C((P \vee Q) \Rightarrow (\neg Q \Rightarrow P))$	4, [C-I]
6.	$\{S_1, S_2, S_3\} \vdash C(P \vee Q) \Rightarrow C(\neg Q \Rightarrow P)$	5, [C_K], [$\Rightarrow$-E]
7.	$\{S_1, S_2, S_3\} \vdash C(\neg Q \Rightarrow P)$	6, 2, [$\Rightarrow$-E]
8.	$\{S_1, S_2, S_3\} \vdash C(\neg Q \Rightarrow P) \Rightarrow C(K\alpha(\neg Q \Rightarrow P))$	[R]
9.	$\{S_1, S_2, S_3\} \vdash C(K\alpha(\neg Q \Rightarrow P))$	8, 7, [$\Rightarrow$-E]
10.	$\{R_1 \wedge K\alpha(\neg Q \Rightarrow P) \wedge R_3\} \vdash Q$	Lemma 6
11.	$\emptyset \vdash (R_1 \wedge K\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow Q$	10, [$\Rightarrow$-I]
12.	$\{S_1, S_2, S_3\} \vdash C((R_1 \wedge K\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow Q)$	11, [C-I]
13.	$\{S_1, S_2, S_3\} \vdash C(R_1 \wedge K\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow C(Q)$	12, [C_K], [$\Rightarrow$-E]
14.	$\{S_1, S_2, S_3\} \vdash C(R_1 \wedge K\alpha(\neg Q \Rightarrow P) \wedge R_3)$	1, 3, 9, Lemma 5, [$\wedge$-I]
15.	$\{S_1, S_2, S_3\} \vdash C(Q)$	13, 14, [$\Rightarrow$-E]

□

Our method can now be expressed as follows:

$\Phi \leftarrow \{S_1^1, S_2^1, S_3^1\}$;
$\Sigma \leftarrow \Phi \vdash S_2^2$;
Use Lemma 7 to derive Σ;
If $n = 1$ **halt**
else
  For $i = 2$ to n do
    begin
      $\Phi \leftarrow \Phi \cup \{S_1^i, S_3^i\}$;
      $\Sigma' \leftarrow \{S_1^i, S_2^i, S_3^i\} \vdash S_2^{i+1}$;
      Use Lemma 7 to derive Σ';
      $\Sigma'' \leftarrow \Phi \vdash S_2^{i+1}$;
      Use the cut on Σ and Σ' to derive Σ'';
      $\Sigma \leftarrow \Sigma''$
    end

The loop variable i ranges over the interval $2, \ldots, n$. For any i in that interval, we write Φ^i and Σ^i for the values of Φ and Σ upon conclusion of the i^{th} iteration of the loop. A straightforward induction on i will establish:

Lemma 8 (Algorithm correctness). *For any $i \in \{2, \ldots, n\}$,*

$$\Phi^i = \{C(M_1 \vee \cdots \vee M_{n+1})\} \bigcup_{j=1}^{i} \{S_1^j, S_3^j\}$$

while $\Sigma^i = \Phi^i \vdash S_2^{i+1}$.

Hence, $\Phi^n = \Omega_{n+1}$, and $\Sigma^n = \Phi^n \vdash S_2^{n+1} = \Omega_{n+1} \vdash S_2^{n+1} = \Omega_{n+1} \vdash C(M_{n+1})$, which is our goal.

It is noteworthy that no such correctness argument is necessary in the formulation of the algorithm as an Athena method, as methods are guaranteed to be sound. Their results are always logically entailed by the assumption base, assuming that our primitive methods are sound (see Chapter 8 of [3]).

4 Athena implementation

In this section we present the Athena encoding of the epistemic logic and our method for solving the generalized version of the wise men puzzle (refer to the Appendix for a brief review of Athena). We begin by introducing an uninterpreted domain of epistemic agents: `(domain Agent)`. Next we represent the abstract syntax of the propositions of the logic. The following Athena datatype mirrors the abstract grammar for propositions that was given in the beginning of Section 2:

```
(datatype Prop
  True
  False
  (Atom Boolean)
  (Not Prop)
  (And Prop Prop)
  (Or Prop Prop)
  (If Prop Prop)
  (Knows Agent Prop)
  (Common Prop))
```

We proceed to introduce a binary relation `sequent` that may obtain between a finite set of propositions and a single proposition:

```
(declare sequent (-> ((FSet-Of Prop) Prop) Boolean))
```

Here `FSet-Of` is a unary sort constructor: for any sort `T`, `(FSet-Of T)` is a new sort representing the set of all finite sets of elements of `T`. Finite sets are built with two polymorphic constructors: the constant `null`, representing the empty set; and the binary constructor `insert`, which takes an element x of sort `T` and a finite set S (of sort `(FSet-Of T)`) and returns the set $\{x\} \cup S$. We also have all the usual set-theoretic operations available (`union`, `intersection`, etc.).

The intended interpretation is that if `(sequent` S P`)` holds for a set of propositions S and a proposition P, then the sequent $S \vdash P$ is derivable in the epistemic logic via the rules presented in Section 2. Accordingly, we introduce axioms capturing those rules. For instance, the conjunction introduction rule is represented by the following axiom:

```
(define And-I
  (forall ?S ?P ?Q
    (if (and (sequent ?S ?P)
             (sequent ?S ?Q))
        (sequent ?S (And ?P ?Q)))))
```

Note that the lowercase `and` above is Athena's built-in conjunction operator, and hence represents conjunction at the metalanguage level, whereas `And` represents the object-level conjunction operator of the epistemic logic.

The cut rule and the common knowledge introduction (necessitation) rule become:

```
(define cut
  (forall ?S1 ?S2 ?P ?Q
    (if (and (sequent ?S1 ?P)
             (sequent (insert ?P ?S2) ?Q))
        (sequent (union ?S1 ?S2) ?Q))))

(define common-intro-axiom
  (forall ?P ?S
    (if (sequent null ?P)
        (sequent ?S (Common ?P)))))
```

The remaining rules are encoded by similar first-order axioms.

We next proceed to derive several lemmas that are useful for the proof. Some of these lemmas are derived completely automatically via the ATPs that are integrated with Athena. For instance, the cut rule is proved automatically (in about 10 seconds). As another example, the following result—part (b) of Lemma 4—is proved automatically:

```
(forall ?S ?P1 ?P2
  (if (and (sequent null (If ?P1 ?P2))
           (sequent ?S (Common ?P1)))
      (sequent ?S (Common ?P2))))
```

Other lemmas are established by giving natural deduction proofs. For instance, the proof of Lemma 6 in Section 3 is transcribed virtually verbatim in Athena, and validated in a fraction of a second. (The fact that the proof is abridged—i.e., multiple steps are compressed into single steps—is readily handled by invoking ATPs that automatically fill in the details.) Finally, we are able to prove Lemma 7, which is the key technical lemma. Utilizing the higher-order character of our encoding, we then define a method `main-lemma` that takes an arbitrary list of agents $[a_1 \cdots a_n], n \geq 1$, and specializes Lemma 7 with $P \mapsto M_{a_1}$, $Q \mapsto M_{a_2} \vee \cdots \vee M_{a_n}$, and $\alpha \mapsto a_1$ (recall that for any agent α, M_α signifies that α is marked). So, for instance, the application of `main-lemma` to the list $[a_1, a_2, a_3]$ would derive the conclusion $\{S_1, S_2, S_3\} \vdash C(M_{a_2} \vee M_{a_3})$, where $S_1 = C(\neg K_{a_1}(M_{a_1}))$, $S_2 = C(M_{a_1} \vee M_{a_2} \vee M_{a_3})$, and

$$S_3 = C(\neg(M_{a_2} \vee M_{a_3}) \Rightarrow K_{a_1}(\neg(M_{a_2} \vee M_{a_3})))$$

We also need a simple result `shuffle` asserting the equality $\Gamma, P_1, P_2 = \Gamma, P_2, P_1$ (i.e., $\Gamma \cup \{P_1\} \cup \{P_2\} = \Gamma \cup \{P_2\} \cup \{P_1\}$).

Using these building blocks, we express the tactic for solving the generalized wise men problem as the Athena method `solve` below. It takes as input a list of agents representing wise men, with at least two elements. Note that the for loop in the pseudocode algorithm has been replaced by recursion.

```
(define (solve wise-men)
  (dletrec
    ((loop (method (wise-men th)
             (dmatch wise-men
               ([_] (!claim th))
               ((list-of _ rest)
                 (dlet ((new-th (!main-lemma wise-men)))
                  (dmatch [th new-th]
                    ([(sequent context Q2)
                      (sequent (insert Q1
                                 (insert Q2 (insert Q3 null))) P)]
                       (dlet ((cut-th
                                (!derive (sequent
                                           (union
                                             context
                                             (insert Q1 (insert Q3 null)))
                                           P)
                                          [th new-th shuffle cut])))
                         (!loop rest cut-th))))))))))
    (dlet ((init (!prove-goal-2 wise-men)))
      (!loop (tail wise-men) init))))
```

Assuming that `w1`, `w2`, `w3` are agents representing wise men, invoking the method `solve` with the list `[w1 w2 w3])` as the argument will derive the appropriate result: $\Omega_3 \vdash$ `(Common (isMarked w3))`, where Ω_3 is the set of premises for the three-men case, as defined in the previous section.

5 Related work

The wise men problem became a staple of epistemic AI literature after being introduced by McCarthy [31]. Formalizations and solutions of the two-wise-men problem are found in a number of sources [27, 40, 20], most of them in simple multi-agent epistemic logics (without common knowledge). Several variations have been given; e.g., Konolige has a version in which the third wise man states that he does not know whether he is marked, but that he would know if only the second wise man were wiser [29]. Ballim and Wilks [9] solve the three-men version of the puzzle using the "nested viewpoints" framework. Vincenzo Pallotta's solution [34] is similar but his ViewGen framework facilitates agent simulation. Kim and Kowalski [28] use a Prolog-based implementation of metareasoning to solve the same version of the problem using common knowledge. A more natural proof was given by Aiello et al. [1] in a rewriting framework.

The importance of reasoning about the intentional states of intelligent agents is widely recognized (see, for instance, the recent work by Dastani et al. on inferring trust [?]). Agent metareasoning and metaknowledge, in particular, is extensively discussed in "Logical foundations of Artifical Intelligence" by Genesereth and Nillson [20] (it is the subject of an entire chapter). They stress that the main advantage of an explicit encoding of the reasoning process is that it makes it possible to "create agents capable of reasoning in detail about the inferential abilities of and beliefs of other agents," as well as enabling introspection.[3]

The only work we are aware of that has an explicit encoding of an epistemic logic in a rich metalanguage is a recent project [30] that uses the Calculus of Constructions (Coq [12]). However, there are important differences. First, they encode a Hilbert proof system, which has an adverse impact on the readability and writability of proofs. The second and most important difference is our emphasis on reasoning efficiency. The seamless integration of Athena with state-of-the-art provers such as Vampire and Spass is crucial for automation, as it enables the user to skip tedious steps and keep the reasoning at a high level of detail. Another distinguishing aspect of our work is our heavy use of tactics. Athena uses a block-structured natural-deduction style not only for writing proofs but also for writing proof tactics ("methods"). Proof methods are much easier to write in this style, and play a key role in proof automation. Our emphasis on automation also differentiates our work from that of Basin et al. [10] using Isabelle, which only addresses proof presentation in modal logics, not automatic proof discovery.

References

1. L. C. Aiello, D. Nardi, and M. Schaerf. Yet another solution to the three wisemen puzzle. In *Proceedings of the 3rd International Symposium on Methodologies for Intelligent Systems*, pages 398–407, 1988.
2. K. Arkoudas. Athena. `http://www.cag.csail.mit.edu/~kostas/dpls/athena`.
3. K. Arkoudas. Denotational Proof Languages. PhD dissertation, MIT, 2000.
4. K. Arkoudas. Specification, abduction, and proof. In *Second International Symposium on Automated Technology for Verification and Analysis*, Taiwan, October 2004.
5. K. Arkoudas, S. Khurshid, D. Marinov, and M. Rinard. Integrating model checking and theorem proving for relational reasoning. In *Proceedings of the 7th International Seminar on Relational Methods in Computer Science (RelMiCS 7)*, Malente, Germany, May 2003.
6. K. Arkoudas and M. Rinard. Deductive runtime certification. In *Proceedings of the 2004 Workshop on Runtime Verification*, Barcelona, Spain, April 2004.
7. K. Arkoudas, K. Zee, V. Kuncak, and M. Rinard. Verifying a file system implementation. In *Proceedings of the 2004 International Conference on Formal Engineering Methods (ICFEM)*, Seattle, USA, November 2004.
8. T. Arvizo. A virtual machine for a type-ω denotational proof language. Masters thesis, MIT, June 2002.
9. A. Ballim and Y. Wilks. *Artificial Believers*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.
10. David Basin, Seán Matthews, and Luca Viganò. A modular presentation of modal logics in a logical framework. In Jonathan Ginzburg, Zurab Khasidashvili, Carl

[3] In addition, Bringsjord and Yang [?] have claimed that the best of human reasoning is distinguished by a capacity for meta-reasoning, and have proposed a theory—mental metalogic—of human and machine reasoning that emphasizes this type of reasoning.

Vogel, Jean-Jacques Lévy, and Enric Vallduví, editors, *The Tbilisi Symposium on Logic, Language and Computation: Selected Papers*, pages 293–307. CSLI Publications, Stanford, CA, 1998.
11. K. Claessen and N. Sorensson. New techniques that improve Mace-style finite model building. In *Model Computation—principles, algorithms, applications*, Miami, Florida, USA, 2003.
12. T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76:95–120, 1988.
13. D. Cyrluk, S. Rajan, N. Shankar, , and M.K. Srivas. Effective theorem proving for hardware verification. In *Theorem Provers in Circuit Design (TPCD '94)*, volume 901 of *Lecture Notes in Computer Science*, pages 203–222, Bad Herrenalb, Germany, September 1994. Springer-Verlag.
14. E. Davis and L. Morgenstern. Epistemic Logics and its Applications: Tutorial Notes. `www-formal.stanford.edu/leora/krcourse/ijcaitxt.ps`.
15. Giunchiglia E., Giunchiglia F., Sebastiani R., and Tacchella A. More evaluation of decision procedures for modal logics. In Cohn A. G., Schubert L., and Shapiro S. C., editors, *6th international conference on principles of knowledge representation and reasoning (KR'98)*, Trento, 2-5 June 1998.
16. H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 2nd edition, 1994.
17. R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. MIT Press, Cambridge, Massachusetts, 1995.
18. M. Fitting. Basic modal logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Logical foundations*, volume 4 of *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford Science Publications, 1994.
19. D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. Many-dimensional modal logics: theory and applications. volume 4 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1994.
20. M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.
21. M. J. C. Gordon and T. F. Melham. *Introduction to HOL, a theorem proving environment for higher-order logic*. Cambridge University Press, Cambridge, England, 1993.
22. J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
23. M. Hao. Using a denotational proof language to verify dataflow analyses. Masters thesis, MIT, September 2002.
24. A. Heuerding. LWBtheory: information about some propositional logics via the WWW. *Logic Journal of the IGPL*, 4(4):169–174, 1996.
25. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Sixth International Conference on Principles of Knowledge Representation and Reasoning*, pages 636–647, 1998.
26. U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Fifteenth International Joint Conference on Artificial Intelligence*, pages 202–209, 1997.
27. M. Huth and M. Ryan. *Logic in Computer Science: modelling and reasoning about systems*. Cambridge University Press, Cambridge, UK, 2000.
28. J. Kim and R. Kowalski. An application of amalgamated logic to multi-agent belief. In M. Bruynooghe, editor, *Second Workshop on Meta-Programming in Logic META90*, pages 272–283. 1990.

29. K. Konolige. *A deduction model of belief*. Research Notes in Artificial Intelligence. Pitman, London, UK, 1986.
30. Pierre Lescanne. Epistemic logic in higher order logic: an experiment with COQ. Technical Report RR2001-12, LIP-ENS de Lyon, 2001.
31. J. McCarthy. Formalization of two puzzles involving knowledge. In Vladimir Lifschitz, editor, *Formalizing Common Sense: Papers by John McCarthy*. Ablex Publishing Corporation, Norwood, New Jersey, 1990.
32. J.J. Meyer and W. Van Der Hoek. Epistemic Logic for Computer Science and Artificial Intelligence. volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.
33. D. Musser. Generic Software Design. `http://www.cs.rpi.edu/~musser/gsd`.
34. V. Pallotta. Computational dialogue Models. In *10th Conference of the European Chapter of the Association for Computational Linguistics EACL03*, 2003.
35. L. Paulson. *Isabelle, A Generic Theorem Prover*. Lecture Notes in Computer Science. Springer-Verlag, 1994.
36. F. J. Pelletier. A Brief History of Natural Deduction. *History and Philosophy of Logic*, 20:1–31, 1999.
37. M. Rinard and D. Marinov. Credible compilation with pointers. In *Proceedings of the 1999 Workshop on Run-Time Result Verification*, Trento, Italy, July 1999.
38. R. A. Schmidt. MSPASS. `http://www.cs.man.ac.uk/~schmidt/mspass/`, 1999.
39. R. A. Schmidt and U. Hustadt. Mechanised reasoning and model generation for extended modal logics. In H. C. M. de Swart, E. Orlowska, G. Schmidt, and M. Roubens, editors, *Theory and Applications of Relational Structures as Knowledge Instruments*, volume 2929 of *Lecture Notes in Computer Science*, pages 38–67. Springer, 2003.
40. D. Snyers and A. Thayse. Languages and logics. In A. Thayse, editor, *From modal logic to deductive databases*, pages 1–54. John Wiley & Sons, 1989.
41. A. Voronkov. The anatomy of Vampire: implementing bottom-up procedures with code trees. *Journal of Automated Reasoning*, 15(2), 1995.
42. C. Weidenbach. Combining superposition, sorts, and splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2. North-Holland, 2001.

A Athena Overview

Athena is a new interactive theorem proving system that incorporates facilities for model generation, automated theorem proving, and structured proof representation and checking. It also provides a higher-order functional programming language, and a proof abstraction mechanism for expressing arbitrarily complicated inference *methods* in a way that guarantees soundness, akin to the tactics and tacticals of LCF-style systems such as HOL [21] and Isabelle [35]. Proof automation is achieved in two ways: first, through user-formulated proof methods; and second, through the seamless integration of state-of-the-art ATPs such as Vampire [41] and Spass [42] as primitive black boxes for general reasoning. For model generation, Athena integrates Paradox [11], a new highly efficient model finder. For proof representation and checking, Athena uses a block-structured Fitch-style natural deduction calculus [36] with novel syntactic constructs and a formal semantics based on the abstraction of *assumption bases* [3]. Most interestingly, a block-structured natural deduction format is used not only for writing proofs, but also for writing tactics (methods). This is a novel feature of Athena; all other tactic languages we are aware of are based on sequent calculi. Tactics in this style are considerably easier to write and remarkably useful in making proofs more modular and abstract.

Athena [2,4] has been used to implement parts of a proof-emitting optimizing compiler [37]; to integrate model checking and theorem proving for relational reasoning [5]; to implement various "certifying" algorithms [6]; to verify the core operations of a Unix-like file system [7]; to prove the correctness of dataflow analyses [23]; and to reason about generic software [33]. This section presents parts of Athena relevant to understanding the code in this paper. A comprehensive tutorial for the language can be found on the Athena web site [2], while a succinct presentation of its syntax and semantics can be found elsewhere [8].

In Athena, an arbitrary universe of discourse (sort) is introduced with a `domain` declaration, for example:

```
(domain Real)
```

Function symbols and constants can then be declared on the domains, e.g.:

```
(declare + (-> (Real Real) Real))
```

Relations are functions whose range is the predefined sort `Boolean`, e.g.,

```
(declare < (-> (Real Real) Boolean))
```

Inductively generated domains are introduced as *datatypes*, e.g.,

```
(datatype Nat
  zero
  (succ Nat))
```

Here `Nat` is freely generated by the *constructors* `zero` and `succ`. When the datatype is defined, a number of axioms and a structural induction principle are automatically generated, constraining `Nat` to freely generated by `zero` and `succ`.

The user interacts with Athena via a read-eval-print loop. Athena displays a prompt `>`, the user enters some input (either a phrase to be evaluated or a top-level directive such as `define`, `assert`, `declare`, etc.), Athena processes the user's input, displays the result, and the loop starts anew.

An Athena deduction D is always evaluated in a given *assumption base* β—a finite set of propositions that are assumed to hold for the purposes of D. Evaluating D in β will either produce a proposition P (the "conclusion" of D in β), or else it will generate an error or will diverge. If D does produce a conclusion P, Athena's semantics guarantee $\beta \models P$, i.e., that P is a logical consequence of β. Athena starts out with the empty assumption base, which then gets incrementally augmented with the conclusions of the deductions that the user successfully evaluates at the top level of the read-eval-print loop. A proposition can also be explicitly added into the global assumption base with the top-level directive `assert`.