Multi-Agent Belief Reasoning in a First-Order Logic Back-Chainer Technical Report

Arnold Binas and Thomas R. Ioerger
{a0b1767, ioerger}@cs.tamu.edu

September 14, 2004

Abstract

This report gives a brief overview of and a few pointers on belief reasoning for multi-agent systems. The modeling of beliefs is usually done in terms of modal logics which are fairly costly to reason in. We introduce our novel, more practical approach to belief reasoning in a first-order back-chaining system such as Prolog. At the end of the report, we justify the approach by relating some of its properties to those of weak-S5, a common modal belief logic, before concluding with a real-world example illustrating the approach.

1 Introduction

Reasoning about other agents' beliefs is widely recognized as an important necessity in multi-agent systems. The motivations for such a capability are fairly obvious. Knowing or reasoning about others' beliefs can benefit an agent both in competitive and cooperative environments. In competitive environments, knowledge about the beliefs of competitors enables an agent to anticipate their actions, which, in turn, helps it in determining its own actions. In a cooperative environment, anticipating teammates' actions can save communication time and enhance collaboration [5, 8].

A good amount of work has been published on belief reasoning for multi-agent systems. However, the vast majority of this work models beliefs in modal logics, which are very powerful and expressive but at the same time very costly to reason in. For example, Halpern and Moses [3] show that determining entailment even in S5, a common modal logic for knowledge, has PSPACE-complexity. Two prominent examples of work modeling belief in modal logics are [2] and [6]. In [2], Cohen and Levesque develop a model of intention based on two separate modalities, beliefs and goals. The authors define the intentions of an agent in terms of its *persistent goals* and persistent goals in terms of its beliefs and goals. In [6], Rao and Georgeff introduce a full belief, desire, and intention (BDI) architecture based on three distinct modalities. In this context, *desire* is synonym for *goal*. Rao and Georgeff describe a formalism that enables agents to reason about the future via a branching model of time. For the abovementioned reasons of complexity of contemporary techniques, we propose a practical approach to belief reasoning that, at the expense of some expressiveness, enables agents to efficiently reason about other agents' beliefs.

In this report, we will introduce our approach to approximating belief reasoning in a first-order logic back-chaining system and demonstrate its usefulness. We then discuss how beliefs are commonly represented in a modal logic and tie this approach to our representation before formally establishing the properties of our approach. A more complex example of belief representation and reasoning in the new approach and a summarizing discussion conclude the report.

2 Belief Reasoning in FOL

In this section, we describe an approach to practical belief reasoning via back-chaining in first-order logic. The approach is motivated by the complexity of conventional belief reasoning based on modal logic. In [3], Halpern and Moses show that deciding satisfiability of a modal logic formula is NP-complete for a single agent and even worse for multiple agents. We introduce our representation with the help of some examples in Prolog, although, of course, the approach also works in any other first-order back-chainer.

In order to reason about the beliefs of others from the perspective of an individual agent, we must be able to specify the belief context as well as the beliefs. The belief context refers to which agent's be-

liefs are considered. Beliefs themselves are generally believed facts. Believed rules are expressed as rules about believed facts in a certain belief context. In Prolog, the *belief operator* is modeled as a predicate with two arguments, the belief context and the belief itself. Rules can operate within a context or bridge between contexts. The belief context is specified as a list of agent identifiers in reverse order.

Let us discuss the details of the representation by the means of an example. Consider the following Prolog statement.

```
bel([bill,steven], light_on(kitchen)).
```

`bel` is the belief predicate, taking two arguments. The first is a list specifying the belief context. The second is the believed fact, i.e. that the light is on in the kitchen. So in this case, Steven believes that Bill believes that the light is on in the kitchen, which is equivalent to the following modal expression.

$$B_{steven}\ B_{bill}\ light\_on(kitchen)$$

Note the reverse order of "believers" in the Prolog list specifying the belief context. The reason for listing believers in reverse order is that it is easy to extract the first member of a list, in our case the immediate believer of the fact in the second argument of the belief predicate. We are thus taking advantage of unification on the first argument and Prolog's list notation to extract a particular believer. For example, `[X|Y]` unifies with `[bill, steven]` with `X` bound to `bill` and `Y` bound to the rest of the list, in this case `[steven]`.

Note how this representation enables one agent to model the mental states of others. I.e., all facts and rules that one agent believes another agent has in its knowledge base form a model of that knowledge base as seen from the believing agent. Since each agent separately reasons about its own and others' beliefs, or models of their knowledge bases, we need some way of referring to the owner of the knowledge base, i.e. the agent that does the reasoning. This is accomplished by using the special identifier `self`. For example, the agent believing p would look like this:

```
bel([self], p).
```

while John believing that the agent believes p would be expressed as follows.

```
bel([self,john], p).
```

So far, we are able to express facts that a certain agent believes, or that it believes that another agent believes, etc. In order to reason about these facts, however, we must also be able to represent believed rules such as in the following example.

$$bel(X, P) \wedge bel(X, Q) \wedge bel(X, P \wedge Q \rightarrow R) \models bel(X, R)$$

This turns out to be somewhat trickier since we cannot wrap the *bel* operator, represented as a predicate, around a Prolog rule and still expect Prolog to do the inferences within a certain belief context. The following pseudo Prolog statement illustrates this observation.

```
bel([bill], (r :- p, q)).  % not a valid Prolog statement
```

The specific rule that p and q imply r can clearly not be placed inside the argument list of the `bel` predicate. To see how we can get around this dilemma, reconsider the following simple rule from the above example.

$$P \wedge Q \rightarrow R$$

If an agent believes this rule to be valid, it should be able to infer believing R when it believes both P and Q. It turns out that we can distribute the belief predicate over implications, conjunctions, and arbitrary combinations thereof. In our example, we arrive at the following rule, solving our dilemma with rules embedded in a Prolog predicate.

```
bel(X, r) :- bel(X, p), bel(X, q)
```

where X is a list of agents. We will formally show in Section 4 that this rule linking different beliefs in fact amounts to believing a rule itself for practical applications. For now, we will rely on our intuition to see why this works by looking at the inference process in the following way. While with one belief operator wrapped around the whole rule, any inferences would be made within a certain belief context, distributing the operator over the rule can be thought of as the inference being made separately and the result being placed back into the original belief context. Note that by using the variable X for the belief context, the rule above becomes *common belief*, i.e. everybody believes that $P \wedge Q \rightarrow R$ and everybody believes that everybody else believes that $P \wedge Q \rightarrow R$, etc. At the same time, it becomes *common knowledge* that everybody believes $P \wedge Q \rightarrow R$, etc. In addition to enabling the expression of common rules within a given context, this

technique also opens a way to reason across different belief contexts. For example, as the `self` agent we might think that John considers us to be very trustworthy and therefore believe that John believes everything that we believe he believes we believe. In our approach, this can be expressed as the following Prolog rule. Recall that belief contexts are given in reverse order in the Prolog lists.

```
bel([john,self], p) :- bel([self,john,self], p).
```

In the following, more complex example, the inference crosses multiple believe contexts.

```
bel([X|Y], light_on(kitchen)) :-
bel([X|Y], in(X, kitchen)), bel(Y, light_on(kitchen)).
```

To illustrate, let us look at the following instantiation of this rule (though it would, of course, rarely be stated like this in a knowledge base). `a` believes that `b` believes that `c` believes the light is on in the kitchen if `a` believes that `b` believes that the light is on in the kitchen and that `c` believes that it is in the kitchen.

```
bel([c,b,a], light_on(kitchen)) :-
bel([c,b,a], in(c, kitchen)),
bel([b,a], light_on(kitchen)).
```

Note that the reverse order of belief context members is essential to being able to express such rules. In the above example, we need to exclude `c` from the belief context in one occasion, which we can do by unifying `[X|Y]` with `[c,b,a]` and extracting `c` as `X`. Extracting `c` by itself would not have been possible by expressing the chain of believers in their original order `[a,b,c]`.

We are now in a position to express beliefs in different contexts and reason about them. We can also reason across different belief contexts. In reasoning about beliefs, however, it is important that we can distinguish between believing something, say p, to be false and not having a belief about p at all. The latter case amounts to p being unknown to an agent. In Prolog, a query about the negation of a fact succeeds if the fact is not in the knowledge base. I.e., the absence of a fact is treated as the fact being false. Intuitively, one could regard the absence of a fact either as it being false or unknown. In our representation of beliefs, we need to be able to distinguish between these

two cases since there will be many situations where one agent simply has no information about what another believes. Therefore, we cannot simply rely on Prolog's default. To better see why, consider the following query, assuming that the knowledge base does not contain the fact `bel([john], p)`.

```
| ?- not bel([john], p).
yes
```

This query succeeds in Prolog. But what does this result mean? It means that John does not believe p but does not say anything about whether John believes p to be false or whether he just does not know the state of p, i.e. regards it as unknown. To solve this problem, we propose two special functions and a default rule. First, we introduce a function `false` and a function `unk` for unknown. These functions are to be used within a belief context as in the following example.

```
bel([bill], false(p)).
bel([bill], unk(q)).
```

The default rule achieves a similar behavior within belief contexts as Prolog's treatment of false facts and looks as follows.

```
bel(X, false(Y)) :- not bel(X, Y), not bel(X, unk(Y)).
```

Or in English, everything within a certain belief context is regarded as believed false if it is neither believed nor explicitly believed unknown in that context. Therefore only unknown facts have to be explicitly stated; beliefs that a fact is false are derived from the lack of explicit belief and unknowness. It could, of course, also be done the other way around, with "false" being explicit and "unknown" the default, if desired.

3 Modal Logic for Belief

In this section, we briefly introduce the commonly taken approach of modeling agents' beliefs in a modal logic. We first review the basics of modal logics, their axiomatizations and semantics, before drawing the connection to belief reasoning.

3.1 Modal Logic

Modal logics operate on top of standard propositional or first-order logic by introducing one or more modal operators. We will discuss *normal modal logics*, on which most modal logics are based. The syntax of normal modal logic is the syntax of its underlying logic (propositional or first-order) plus a few rules to define the syntax of modal operators. If ϕ and ψ are formulae (in the sense of the underlying logic), then so are $\neg\phi$, $\phi \vee \psi$, $\Box\phi$, and $\Diamond\phi$. The symbols $\Box$ and $\Diamond$ are read *necessarily* and *possibly*, respectively, and are the modal operators. We will see later that one can be defined in terms of the other and there is therefore only one unique modality in normal modal logics. The semantics of modal operators are captured in the *possible-worlds model* first introduced by Hintikka in [4]. In possible-worlds semantics, a state is characterized by a number of *accessible possible worlds*. Each such world represents a state of the propositional (or first-order) formulae in the theory. What propositional states (worlds) are accessible then determines the truth value of the modal formula. What kinds of worlds are accessible, in turn, depends on the current state as determined by the interaction of all facts and rules in the knowledge base.

We are now ready to define the semantics of the *necessarily* and *possibly* operators. The formula $\Box\phi$ is true if and only if ϕ is true in all accessible worlds given the current state. Or, more formally,

$$\langle M, w\rangle \models \Box\phi \text{ } \textit{if and only if } \forall w' \in W \textit{ if } (w, w') \in R \textit{ then } \langle M, w'\rangle \models \phi,$$

where M is a model, w and w' are worlds, W is the set of all possible worlds, and R is the accessibility relation of the $\Box$ operator [7, page 272]. For example, $\Box raining$ is true iff it is raining in all accessible possible worlds. As mentioned above, what worlds are accessible depends on the current state of the world. Likewise, the formula $\Diamond\psi$ is true if and only if ψ is true in at least one accessible world in the current state. Now we can actually see how one of the two normal modal operators can be defined in terms of the other. Consider a situation in which $\Diamond\phi$ is true. That is, in some, but not necessarily all, accessible worlds, ϕ is true. Another way of describing the same situation is to say $\Box\neg\phi$ is false or $\neg\Box\neg\phi$ is true, leaving room for accessible worlds in which ϕ is true. Therefore we can write

$$\Diamond\phi \Leftrightarrow \neg\Box\neg\phi,$$

which is equivalent to the statement

$$\Box\phi \Leftrightarrow \neg\Diamond\neg\phi.$$

Properties of the accessibility relation correspond to axioms that are an alternative way of stating the characteristics and effects of the modal operators. Let us look at five axioms, any combination of which can be assumed for a normal modal logic (Table 1).

Table 1: The five axioms of normal modal logics

Name	Axiom	Accessibility Relation Property
K	$\Box(\phi \rightarrow \psi) \models (\Box\phi \rightarrow \Box\psi)$	
T	$\Box\phi \models \phi$	Reflexive
D	$\Box\phi \models \Diamond\phi$	Serial
4	$\Box\phi \models \Box\Box\phi$	Transitive
5	$\Diamond\phi \models \Box\Diamond\phi$	Euclidean

As stated above, each axiom corresponds to a property of the accessibility relation in the possible-worlds model. The correspondences are listed in Table 1 along with the axioms and the reader can convince herself or consult more elaborate introductions to modal logics (such as [1]) to see why these correspondences are true.

3.2 Knowledge and Belief for a Single Agent

Different combinations of these axioms give different modal logics. The classic modal logic with the necessity and possibility operators results from a combination of all five axioms, K, T, D, 4, and 5 and is also known as S5. We will briefly examine this logic, as it and a weaker form of it are of major importance to reasoning in multi-agent systems. In S5, the $\Box$ operator can be used as an operator *knows*, resulting in a logic of knowledge (or *epistemic* logic) for an agent. In a logic of knowledge, the $\Diamond$ operator has no defined meaning and we write all occurrences of it in terms of the $\Box$ operator as seen above. The first axiom, K, now means $knows\ (\phi \rightarrow \psi) \models knows\ \phi \rightarrow knows\ \psi$. I.e., if an agent knows that ϕ implies ψ, then it automatically knows ψ if it knows ϕ. In terms of our logic of knowledge, this, of course, makes a lot of sense. The next axiom, T, is perhaps the most distinguishing in our logic of knowledge, as we shall see in a moment. It states that

$K\phi \models \phi$, or if an agent knows ϕ then ϕ must be true. We would not call it *knowledge* if an agent could "know" something without it being true. The D axiom, in terms of only the $\Box$ operator, states that the agent cannot know *not* ϕ if it knows ϕ. In other words, the D axiom ensures consistency of knowledge. Axiom 4, also known as positive introspection, states that an agent who knows ϕ also knows that it knows ϕ. Writing axiom 5 in terms of only the $\Box$ operator (meaning "knows" in our logic) results in the following statement.

not knowing $\neg\phi$ *entails knowing that one does not know* ϕ

In other words, an agent reasoning in S5 is aware of its own lack of knowledge. This property is known as *negative introspection*. It should be stated that S5, based on the five axioms mentioned, is equivalent to $KT5$ as the D and 4 follow from the remaining three.

Modifying the axiom base slightly by removing the T axiom results in a modal logic of belief (or *doxastic* logic). This logic, based on the axioms K, D, 4, and 5, is called weak-S5 and resembles S5 in all ways except that it does not enforce the T axiom. Recall that the T axiom states that everything known must be true. In our logic of belief, the $\Box$ operator means *believes* and the T axiom would amount to saying everything believed must be true. This is clearly not our perception of belief, therefore the T axiom is not part of the belief logic. The D axiom suits our purpose better, as it states that

$$B\phi \models \neg B\neg\phi$$

where we have again substituted the definition of the $\Diamond$ operator in terms of the $\Box$ operator (or B for "believe" in this case). Therefore, in weak-S5, an agent is able to believe anything as long as it does not also believe the opposite, ensuring the consistency of beliefs.

While the possible-worlds model along with its corresponding axiomatization is a very elegant way of defining semantics for modal logics, there is one important drawback known as the logical omniscience problem. The problem is that agents employing modal logics are implied to be perfect reasoners by the axioms, i.e. they are logically omniscient [7, page 276]. For reasons discussed in more detail in [7], a modal belief logic has the following disadvantages (from [7, page 277]):

- agents believe all valid formulae;
- agents' beliefs are closed under logical consequence;

- equivalent propositions are identical beliefs; and
- if agents are inconsistent, then they believe everything.

However, the modal logic weak-S5 is still a very useful approximation of belief as in practice. All agents are necessarily bounded reasoners, which are limited to tractable inferences and therefore cannot possibly be logically omniscient.

3.3 Knowledge and Belief for Multiple Agents

The modal logic for belief that we have built up in this section so far is able to describe and reason about the beliefs of a single agent. In a multi-agent system, however, we want to be able to reason about the beliefs of several agents. A simple extension serves this purpose. If there are n agents, we introduce n indexed belief operators Bel_i, one for each agent (sometimes called a multi-modal logic). Each of these operators has its own accessibility relation of possible worlds. The resulting new logic is capable of reasoning about the beliefs of as many agents as desired. Note that, again, the T axiom makes an important distinction between a multiple-agent logic of knowledge versus one for belief. Assuming the T axiom,

$$K_a \; K_b \; \phi \models K_a \; \phi,$$

results in agent a knowing everything it knows agent b knows. This property is known as the transmissibility of knowledge. We clearly do not want belief to be transmissible as well, meaning that agent a would believe everything that it believes agent b believes. Therefore, again, we do not assume the T axiom in our (multi-)modal logic of belief.

In the following section we will justify our first-order logic approach to belief reasoning, presented in Section 2 of this report, on the basis of the weak-S5 modal logic for belief.

4 Properties of the Representation

In Section 2 we have used several properties of our belief representation approach without proving their validity. In this section, we formally derive these properties and discuss limitations of the representation.

4.1 Decomposition Lemmas

Properties of major importance are the decomposition lemmas that enable us to express believed rules in Prolog:

$$B_a B_b \ldots B_n(\phi \wedge \psi) = Bel(X, \phi \wedge \psi) \models Bel(X, \phi) \wedge Bel(X, \psi)$$
$$B_a B_b \ldots B_n(\phi \rightarrow \psi) = Bel(X, \phi \rightarrow \psi) \models Bel(X, \phi) \rightarrow Bel(X, \psi),$$

where X is an arbitrarily nested belief context (believers $a \ldots n$). These are the very properties that enable us to model belief reasoning in a first-order back-chainer. We will prove these statements in two steps. First, we will show that a *believed* conjunction or implication can be decomposed into a conjunction or implication *about* two beliefs in a singular belief context. Then we will show how such a decomposition can be applied iteratively to decompose believed rules in arbitrarily nested belief contexts. Note that we look at Bel as a modal operator in this treatment as our goal is to reduce it to a first-order predicate for use in Prolog.

The decomposition of conjunctions in a singular belief context can be verified via the possible-worlds model of the Bel operator's semantics. Note that we borrow Prolog's list notation to specify the belief context. While it is singular for now (i.e. one believer agent A, nesting one level deep), this notation will help us in proving Lemma 3 when we have to deal with arbitrarily nested belief contexts.

Lemma 1—Decomposition of Conjunctions in a Singular Belief Context. $Bel([A], \phi \wedge \psi) \models Bel([A], \phi) \wedge Bel([A], \psi)$.

Proof. Let M be a model, w and w' worlds, W the set of all possible worlds, and R_A the accessibility relation for agent A's belief operator.

$$\begin{aligned}
&\langle M, w\rangle \models Bel([A], \phi \wedge \psi) \\
&\Leftrightarrow \forall w' \in W \cdot \textit{if } (w, w') \in R_A \textit{ then } \langle M, w\rangle \models \phi \wedge \psi \\
&\Leftrightarrow \forall w' \in W \cdot \textit{if } (w, w') \in R_A \textit{ then } \langle M, w\rangle \models \phi \wedge \langle M, w\rangle \models \psi \\
&\Leftrightarrow Bel([A], \phi) \wedge Bel([A], \psi).
\end{aligned}$$

□

Therefore, a conjunction believed in a singular belief context can be decomposed into a conjunction of individual beliefs, each in the same singular belief context.

We now need to verify that the same kind of decomposition also holds for believed implications.

Lemma 2—Decomposition of Implications in a Singular Belief Context. $Bel([A], \phi \rightarrow \psi) \models Bel([A], \phi) \rightarrow Bel([A], \psi)$.

Proof. This is the K axiom with A's belief operator substituted for the $\Box$ operator: $\Box(\phi \rightarrow \psi) \models \Box\phi \rightarrow \Box\psi$.

□

By similar reasoning, the results in Lemmas 1 and 2 can be combined to decompose complete believed rules consisting of conjunctions and implications.

It remains to be shown how rules that are believed in nested belief contexts can be reduced to the case of a singular context. As an example, we will demonstrate a proof for decomposing the simple believed rule $\phi \wedge \psi \rightarrow \chi$. The result also holds for arbitrary combinations of conjunctions and implications by similar reasoning.

Lemma 3—Decomposition of Rules in Nested Belief Contexts. $Bel(X, \phi \wedge \psi \rightarrow \chi) \models Bel(X, \phi) \wedge Bel(X, \psi) \rightarrow Bel(X, \chi)$, *where X is a belief context specified by a chain of n believers.*

Proof. This result is proved by induction on the depth n of the belief context X.

Basis
$Bel([A], \phi \wedge \psi \rightarrow \chi) \models Bel([A], \phi) \wedge Bel([A], \psi) \rightarrow Bel([A], \chi)$,
where $[A]$ is a belief context of depth $n = 1$, i.e. an individual agent. The base case is true by a combination of the results in Lemmas 1 and 2 as discussed above.

Inductive Hypothesis
Assume the lemma holds for $k - 1$ believers (i.e. nesting $k - 1$ levels deep). We now have to show that it holds for k believers.

Proof of Inductive Step
Let $[H|T]$ be the nested belief context. Recall that the belief context is specified in reverse order, i.e. H is the immediate and individual believer of the rule and T is a list of the chain of all outer believers.

Then

$$
\begin{aligned}
& Bel([H|T], \phi \wedge \psi \rightarrow \chi) \\
& = Bel(T, Bel([H], \phi \wedge \psi \rightarrow \chi)) \\
& = Bel(T, Bel([H], \phi) \wedge Bel([H], \psi) \rightarrow Bel([H], \chi)) \\
& = Bel(T, Bel([H], \phi)) \wedge Bel(T, Bel([H], \psi)) \rightarrow Bel(T, Bel([H], \chi)) \\
& = Bel([H|T], \phi) \wedge Bel([H|T], \psi) \rightarrow Bel([H|T], \chi).
\end{aligned}
$$

On line 4 we apply Lemmas 1 and 2, the equality on line 5 uses the iductive hypothesis. Note that this argument also holds if the original rule already is a rule about beliefs. In this case, the "new" believer H is simply added to their respective belief contexts.

Conclusion
Therefore, a rule believed *in* a nested belief context can be expressed as a rule *about* beliefs with equally deeply nested belief contexts.

□

4.2 Limitations

Of course, the representation of what is really a modal logic in a first-order back-chainer also comes at some cost (due to the tradeoff between expressiveness and tractability). First, the representation is clearly only capable to handle conjunctive rules; the decomposition lemmas cannot be applied to disjunctions like $Bel_a\ Bel_b\ p \vee q$. Another notable restriction of the approach is that negative beliefs can only be represented in the deepest level of belief. To illustrate this fact, consider the following expression.

$$Bel_a\ \neg Bel_b\ Bel_c\ \phi$$

The above is not expressible in our approach due to the Prolog list representation of nested believers. In our approach, negative beliefs are expressed using the `false` function, which can only act on a fact believed to be false. The belief context is captured in the list in the `bel` predicate's first argument and individual elements of it are thus not accessible to negation. However, note that the deepest level of belief can still be negated, i.e. a fact can be believed to be false in a given belief context. According to our representation of belief contexts introduced in Section 2, the Prolog statement

```
bel([c,b,a], false(p)).
```

denotes

$$Bel_a\ Bel_b\ Bel_c\ \neg p.$$

Also expressible in our representation is the following.

$$Bel_a\ Bel_b\ \neg Bel_c\ p.$$

a believes that b believes that c does not believe p. Recall that the default rule we introduced in Section 2 states that everything neither believed nor believed unknown is believed to be false. Therefore, we can express the above example in Prolog as follows.

```
bel([c,b,a], unk(p)).
```

If `a` believes that `b` believes that `c` believes `p` to be unknown, then, by the meaning of the `unk` function, `a` believes that `b` believes that `c` does not believe `p`.

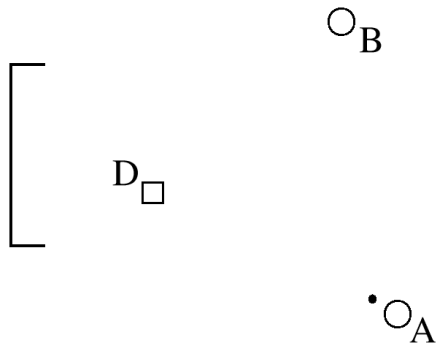
5 An Example

In this section, we illustrate our approach to practical belief reasoning by the means of a more thorough example. The example demonstrates the most important properties and capabilities of the representation introduced in this report. It involves three agents who are situated in a partially competitive and partially cooperative environment. The following subsections discuss the situation first in general and then in terms of concrete Prolog rules and queries demonstrating the workings of belief reasoning in a first-order logic back-chainer.

5.1 The Situation

The example considered here is a give-and-go situation in a soccer game. Two players of the attacking team, one of which is in possession of the ball, have a defender of the opposing team between them and the goal and want to shoot the ball into the goal (Figure 1). Depending on what action the defender takes (guard the player without the ball or attack the player with the ball), the attacking player with the ball must either directly go for the goal and shoot or initiate the give-and-go play by passing the ball to its teammate. Once the play is initiated the player receiving the pass must either shoot or pass the ball back, again depending on the actions of the defender.

Figure 1: A give-and-go situation in soccer



5.2 A Prolog Model of the Situation

We will use the functions in Table 2 to describe actions and states of the world.

Table 2: Prolog functions for the soccer domain

Function	Meaning
`ball(X)`	X is in possession of the ball
`move(X,Y)`	X moves towards Y
`stay(X,Y)`	X stays by Y, i.e. no movement detected
`shoot(X)`	X shoots the ball/moves to shoot
`pass(X,Y)`	X passes the ball to Y
`receive(X,Y)`	X receives pass from Y
`do_nothing(X)`	X does nothing

Any agent can access this information as it is assumed to be directly observable in the soccer domain. The states and actions listed in Table 2 can therefore be considered to be common knowledge. As discussed throughout this report, the predicate `bel` is used to describe beliefs. We can now describe the formation of beliefs based on observations. For example, if Player A has the ball and the defender is approaching him, Player A would form the following three beliefs, where A and B are the attacking players and D is the defender on the opposing team.

- A believes that D believes A is going to shoot (why else should D approach him).
- A believes that B believes A is going to pass the ball to him (both players assume they are playing the same play, i. e. give-and-go).
- A believes that B is ready to receive the ball from him (by the same argument as above).

Note that each of these formed belief rules assumes common knowledge of the state of the world and the actions by the other agents involved. Figure 2 lists belief formation rules for all the possible situations described in Section 5.1. Note that, in this example, we are not using the `self` identifier to refer to a specific agent doing the reasoning. Instead the variable `A` is used to keep the rules general. While the variable name `A` is used to aid understanding of the rules from the perspective of player A, through unification, the rules can be used by any agent on the attacking team. The self reference is thus implicitly used once unification ties a rule to its owner.

Figure 2: Prolog rules for belief formation

```
% D moves towards B and A has the ball
bel([D,A], pass(A,B)) :- move(D,B), ball(A).
bel([B,A], shoot(A)) :- move(D,B), ball(A).
bel([A], do_nothing(B)) :- move(D,B), ball(A).

% D moves towards A who has the ball
bel([D,A], shoot(A)) :- move(D,A), ball(A).
bel([B,A], pass(A,B)) :- move(D,A), ball(A).
bel([A], receive(B,A)) :- move(D,A), ball(A).

% D stays by A and B has the ball
bel([D,A], pass(B,A)) :- stay(D,A), ball(B).
bel([B,A], do_nothing(A)) :- stay(D,A), ball(B).
bel([A], shoot(B)) :- stay(D,A), ball(B).

% D moves towards B who has the ball
bel([D,A], shoot(B)) :- move(D,B), ball(B).
bel([B,A], receive(A)) :- move(D,B), ball(B).
bel([A], pass(B,A)) :- move(D,B), ball(B).
```

Once the agent has beliefs about what is going on as well as about the other agents' beliefs, it can take actions based on these beliefs. In a more complex example, intermediate layers of deliberation could be used, such as desires and intentions. We will, however, limit ourselves to beliefs in this report. Consider Figure 3 which lists Prolog rules for acting based on the beliefs just formed. The first rule, for instance, states that A should shoot if it believes the following three things.

- A believes that D believes that A will pass the ball to B.
- A believes that B believes that A will shoot.
- A believes that B will do nothing.

Figure 3: Prolog rules for acting on beliefs

```
shoot(A) :- bel([D,A], pass(A,B)), bel([B,A], shoot(A)),
bel([A], do_nothing(B)).

pass(A,B) :- bel([D,A], shoot(A)), bel([B,A], pass(A,B)),
bel([A], receive(B,A)).

do_nothing(A) :- bel([D,A], pass(B,A)), bel([B,A],
do_nothing(A)), bel([A], shoot(B)).

receive(A,B) :- bel([D,A], shoot(B)), bel([B,A], receive(A)),
bel([A], pass(B,A)).
```

The reader may have noticed that, in this example, the beliefs are merely a layer between the observed world state and the actions taken. However, this is exactly what beliefs are meant to be. In theory, every world state could be directly mapped to corresponding actions. This, of course, is practically impossible in large domains, and beliefs are one approach to simplify the mapping from world states to actions by one or more intermediate layers. Thus, while in our particular example (due to its simplicity), beliefs do not seem to bring any gain over mapping world states directly to actions, this is not true in general. The purpose of this example is merely to show how belief reasoning can be approached in first-order back-chainer such as Prolog.

5.3 Reasoning About the Situation

Having modeled the give-and-go situation of a soccer game in Prolog, it is time to check if the belief-based decision making actually works. As an example, let us look at the situation where A has the ball and D moves towards B. From the Prolog rules developed above, A should go for the goal and shoot. Asserting `ball(a)` and `move(d,b)` and querying for `shoot` gives

```
| ?- shoot(X).
X = a ?
yes
```

This is what we expected. Behind the scenes, of course, the back-chaining mechanism also derives other, intermediate facts along the way. Let us follow Prolog along one branch in proving our query `shoot(X)`. Prolog starts by finding a rule with our query as the consequence and tries to prove all of its antecedents. In our example, there is one such rule.

```
shoot(A) :-
bel([D,A], pass(A,B)),
bel([B,A], shoot(A)),
bel([A], do_nothing(B)).
```

When trying to prove the first antecedent, Prolog back-chains on the following rule, whose consequence follows immediately from the two facts we initially asserted.

```
bel([D,A], pass(A,B)) :- move(D,B), ball(A).
```

Thus, Prolog derived `bel([D,A], pass(A,B))` as an intermediate fact. The other antecedents of the shoot rule are proven similarly.

In more complex domains, it is often useful to infer new beliefs from existing ones before deciding for an action. In the case where Player D approaches Player A who has the ball, for example, Player A might infer from his belief that Player B believes A is going to pass the ball to him, that Player B also believes that A believes B is ready to receive the pass:

```
bel([A,B,A], receive(B,A)) :- bel([B,A], pass(A,B)).
```

One can also imagine situations in which a soccer agent bases its action or non-action on the believed falseness or unknowness of facts. Consider Player A having the ball and Player B being guarded by the defender. In this situation, Player B might have the belief that A will not pass the ball to him, or

```
bel([B], false(pass(A,B))).
```

Player A, in turn, could be aware of B's belief and basing his decision not to pass, i.e. to shoot, on B's belief that he will not pass the ball. Such a rule would look as follows.

```
shoot(A) :- bel([B,A], false(pass(A,B))).
```

6 Conclusion

In this report, we have argued for the need of efficient belief reasoning techniques for practical purposes. We have introduced such a technique for a first-order logic back-chainer and illustrated it by the means of stand-alone rules and a more thorough, real-world example in Prolog. Furthermore, we have shown where the ideas for the approach come from by reviewing contemporary belief reasoning techniques in modal logic. We also proved the most essential properties of our representation as well as showed some of its weaknesses.

In the future, we will further investigate the potential of this technique and hope to be able to extend it by goals and intentions in the spirit of BDI. Another area we plan to investigate with regards to possibilities for further development of the approach is prioritized belief updating in cooperative environments.

References

[1] Brian F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press: Cambridge, 1980.

[2] Philip R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[3] J. Y. Halpern and Y. Moses. A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief. *Artificial Intelligence*, 54:319–379, 1992.

[4] J. Hintikka. *Knowledge and belief: an introduction to the logic of two notions*. Cornell University Press: New York, 1962.

[5] Thomas R. Ioerger. Reasoning about Beliefs, Observability, and Information Exchange in Teamwork. In *17th International Conference of the Florida Artificial Intelligence Research Society (FALIRS'04)*, 2004.

[6] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.

[7] Michael Woolridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, Inc., 2002.

[8] R.A. Volz Yu Zhang, T.R. Ioerger and J. Yen. Decision-theoretic approach for designing proactive communication in multi-agent teamwork. In *ACM Symposium on Applied Computing, special track on Agents, Interactions, Mobility, and Systems*, pages 64–71, Nicosa, Cyprus, March 14–17 2004.