Reasoning about Beliefs, Observability, and Information Exchange in Teamwork

Thomas R. Ioerger
Department of Computer Science
Texas A&M University
College Station, TX 77843
ioerger@cs.tamu.edu

Abstract

Communication is an important aspect of teamwork, both in human teams and in multi-agent teams. One of the most vital roles for communication is for information exchange, such as for performing team situation assessment. In previous work, we have shown how agents can automatically generate messages for proactive information exchange by inferring relevance based on analysis of requirements of tasks other team members are involved in. However, for the sake of efficiency, it is important to restrict message passing to cases where one agent is reasonably sure another agent does not already believe the information about to be sent. This requires being able to infer and track the belief states of other members on the team. One way this can be done is by reasoning about commonly observable information in the environment. In this paper we introduce a formal framework for reasoning about observability, along with practical algorithms for updating beliefs about other agents' beliefs. We demonstrate the utility of this approach to intelligently filtering communication in a synthetic task domain.

1 Introduction

Communication is a key component of effective teamwork, both in human teams and agent-based teams in multi-agent systems [Tambe, 1997]. One important role for communication in human teamwork is that it can be used to request or offer information or help. Another important role for communication in many human teams, especially those involving tactical decision-making is to build and maintain situation awareness, such as by assimilating information from disparate sources to abstract, classify, and understand what is going on (i.e. information fusion). Examples of this have been observed in teams from sports to business to fire fighting to military tactical operations centers.

Communication is also critical to synthetic teams in multi-agent systems. Communication can be used for task distribution and resource allocation (e.g. via contract nets), planning, information gathering, load balancing, synchronization, coordination, role selection, and joint decision-making (e.g. consensus formation, voting). Communication for negotiation and market-based methods such as auctions are also used in many MAS systems, but not so much in domains with well-structured teams.

Another role for communication in teams is for information sharing, or proactive information exchange. In proactive information exchange, information flows in the opposite direction, compared to information gathering; that is, agents with useful information autonomously forward it to others on the team. This type of information sharing can greatly improve the efficiency of the team's operations. For example, if some new piece of critical information becomes known to one of the agents (such as a new target on the scope of a radar operator on a battleship), or a tank getting low on fuel or ammo just as it is about to head over a hill into battle, it would be prudent to alert the rest of the team. In fact, this may be an assigned role, to watch for information of a particular type and inform the team when it changes. This is especially useful for information that changes infrequently, with regard to the policy of whether the needing agent should Ask or the providing agent should $Tell$[1]; it is more efficient for providing agents to "push" this information only when it changes in these rare situations. However, proactive information exchange can also happen in an unplanned way, where agents, if they are intelligent enough, can infer what new information they observe that might be useful to others, based on knowledge of their current activities. This makes the team smarter than just a collection of individuals (i.e. the "team mind" - acting as if the sensors were centralized, rather than distributed), though it requires each member to know something about the roles and goals of other team members.

In a multi-agent teamwork simulation environment, the generation of proactive information exchanges can be based on the following principle: *the relevance of information to a teammate can be inferred by reasoning about the goals of the other agents, based on their role in the team plan.* Specifically, the pre-conditions of those goals constitute the information relevant to the other agent, since that agent would need to know that information in order to execute the operators. This has also been recognized by [Grosz and Kraus, 1999], who

[1] $Tell$ and *AskIf* are standard KQML performatives [Finin et al., 1997]; the equivalent message types in the FIPA ACL would be $Inform$ and $QueryIf$, respectively, see `www.fipa.org`

discuss the need to communicate in order to keep mutual beliefs about goal achievement and capability up-to-date while executing shared plans.

However, determination of relevance is only half of the criterion for proactive information exchange. The other necessary component is *need*. Agents should only send each other information when they are reasonably sure the other agents do not already know it. Obviously, it is inefficient to keep telling an agent something repetitively just because it is relevant to them; once is enough, then we know that they know it. But there are more complex cases, such as when we realize the other agents can infer it from other information they have. A preliminary formal definition for the criterion for when agent A should send agent B information I proactively is:

Conditions for Proactive Information Exchange:
A should Send message I to B when:

$$Bel(A, I) \wedge Bel(A, \neg Bel(B, I)) \\ \wedge Bel(A, Goal(B, G)) \\ \wedge[(\neg Bel(B, I) \rightarrow \Box \neg Done(B, G))] \\ \wedge[Bel(B, I) \rightarrow \neg \Box \neg Done(B, G)]$$

where $\Box$ is the temporal operator meaning "always."

A concrete example of such a piece of relevant information would the pre-condition of an action that could achieve G. For example, if X is an action, and $I \in PreCond(X)$ and $G \in PostCond(X)$, then I is a prime candidate for information exchange, because agent B would have to know I before being able to execute X to accomplish G; not knowing I would prevent B from ever acting and hence achieving his goal.

This formal criterion can be used as a filter to cut down on the redundant messages, which can be especially important in mixed human-agent teams to keep from overwhelming the human with high volume of either irrelevant or superfluous message traffic from the agent team members.

One important method for inferring what other agents believe is by reasoning about observability in the environment [Isozaki and Katsuno, 1996]. There are many cases in which agents see what each other see (more generally, sense), and from this make inferences about what they believe, which can be used to eliminate the need for certain communications (proactive information exchanges). For example, consider the following example of a team-plan for two agents to capture a burgler. First, agent A will throw open the door and turn on the light, and then agent B will jump on the burgler. Due to the sequential nature of this plan, some synchronization is needed. When agent A sees that the light is on, he will infer that this is now relevant to agent B, who needs the light in the next step of the plan in order to jump on the burgler. But, despite the relevance, agent A should not bother telling agent B that the light is on, as it should be obvious that agent B knows this since he can see the light too. There are many other cases in teamwork simulations where similar situations of mutual observability occur (often but not always involving co-location). In such cases, communication can be obviated through common signals in the environment.

2 Representing Observability

In a teamwork modeling language, we want to be able to describe conditions of observability for the various team members. We view observability as a 3-way relation among: an agent α, a fact ϕ, and a condition ψ specifying a context, which we write as a modal operator: $Obs(\alpha, \phi, \psi)$. The intuitive meaning of $Obs(\alpha, \phi, \psi)$ is that agent α would be able to observe ϕ whenever it was in a situation satisfying ψ. In general, ϕ could be an arbitrary sentence in first-order logic (making Obs modal), though only literals (positive or negative) are handled in our implementation. ψ acts as a context, since agents might only be able to observe certain information under a particular set of circumstances. For example, we might want to say that a scout can see whether there is any enemy on a distant hill provided: a) it has binoculars, b) it is in range, and c) it is daytime. In principle, these could also be captured as antecedents in a rule: $\psi \rightarrow Obs(\alpha, \phi)$. In practice, it should be noted that ϕ and ψ often take α as a parameter, such as that an agent can see whether something it is holding is broken, $Obs(\alpha, broken(X), holding(\alpha, X))$.

As suggested by the example above, our intended interpretation of Obs is unusual in that we take $Obs(\alpha, \phi, \psi)$ to mean not that α knows ϕ specifically, but that α would know *whether* ϕ were true. This is an essential difference. The semantics of the former interpretation can be captured in VSK logic [Wooldridge, 2000] as the S operator, which can be used to indicate facts or expressions which, when true in the real world, can be sensed by an agent. For example, one might want to say that, if the light is on, the agent will sense it: $S_\alpha(lightOn)$. In many applications, agents will believe what they sense, so $S(\phi) \rightarrow K(\phi)$ is often taken as an axiom, where K is the modal operator for knowledge (true belief).

However, we argue that this form of observability is in fact too strong for some applications. It is often convenient to reason about whether an agent believes a fact ϕ without actually committing to specific knowledge of the current truth-value of ϕ. For example, if agent A watches agent B go into a room, agent A will believe that agent B knows whether the light is on in the room, even though agent A might not know himself. This is especially useful in many real-world teamwork applications. For example, in order to figure out which teammate to ask for information, team members need to reason about who knows whether a given fact is true. In some cases, one particular team member might be assigned the responsibility of monitoring information using special sensors or a vantage point. The others need to know that he has information (e.g. whether the fuel in the tank is getting low) without having to know ahead of time the status of that information, i.e. the answer (true or false).

In order to give a semantics to our new observability expression, however, we still base it on VSK logic [Wooldridge, 2000]. VSK is a modal logic with operators for visibility (V), sensability (S), and knowledge (K). These modal operators can be applied to sentences. For example, $V(\phi)$ means that the information ϕ is accessible in the environment, that is, it is possible in principle for agents to distinguish between states in which ϕ is true or false (though whether they do so depends on their sensors; this is not the usual notion of "vis-

ibility" – more like "knowability"). In contrast, $S(\phi)$ means that they actually perceive ϕ to be true, and $K(\phi)$ means that they currently believe it (correctly).

The semantics of VSK logic is based on Kripke-style possible worlds. A space of Kripke structures ("worlds") is defined, each of which encodes the instantaneous state of the environment plus the internal states local to each agent. Then several equivalence relations are used to capture the meanings of the modal operators. For example, there is a relation $\sim_v$ that determines, for each world, what other worlds are indistinguishable (V), and similarly for S and K. The content of what an agent knows is determined by these equivalence relations. For example, an agent is said to believe ϕ if ϕ is satisfied by all the worlds reachable via $\sim_k$ from the current world. If there were a world within the same class of this partition in which $\neg\phi$ were true, this would represent uncertainty/ambiguity, and the agent would not be said to believe ϕ since it considers $\neg\phi$ to be possible. Wooldridge goes on to prove some properties about the inter-relations among V, S, and K in this system, and also offers a proof theory with a guarantee of completeness.

Our observability expression can be modeled based on VSK logic. It is similar to the S operator, though we do not want to say $Obs(\alpha, \phi) \equiv S_\alpha(\phi)$ directly, since this is too strong. Instead of saying that the agent can sense that ϕ is true, we want to say that the agent can sense whether ϕ is true. Therefore, we map our observability expression $Obs(\alpha, \phi)$ into the following:

$$Obs(\alpha, \phi, \psi) \equiv \psi \rightarrow [(\phi \rightarrow S_\alpha(\phi)) \wedge (\neg\phi \rightarrow S_\alpha(\neg\phi))]$$

Hence, according to this definition, the agent will also be able to sense whether ϕ is false too.

We also allow perceptions to influence agents' beliefs, which is ultimately what affects their behavior. Wooldridge et al. suggest that $S(\phi) \rightarrow K(\phi)$ is an axiom trusting agents could typically adopt. This might or might not be a valid assumption, depending on the presence of illusions, sensors faults, etc. in the domain. We feel such a simplification is adequate for our applications of interest, at least for the *self* agent (one can often do no better than believing one's own perceptions, unless using more sophisticated Bayesian techniques...). However, we do need to be able to represent states of belief for other agents that might differ from our view of reality. For example, we might want to represent that another agent believes a light is off even when we believe it is on. Therefore, we adopt the assumption that agents $believe$ what they see (sense): $S(\phi) \rightarrow B(\phi)$, noting that B has the standard epistemic interpretation of a belief state without the requirement of being consistent with the true world, i.e. $B(\phi) \not\rightarrow \phi$, as would be for the K operator. Thus if an agent can sense whether something is true or false, then it will $believe$ whether it is true or false.

3 Modeling Belief States

In this section, we present a practical algorithm for agents to update their beliefs about each other in a dynamic environment that takes into account observability, as well as other types of inference. The main insight is that, when one agent A believes that another agent B has entered a state satisfying the context conditions for some observability expression, A can believe that B believes whether the corresponding fact is true; furthermore, if A himself believes the fact to be true or false, then he may transfer this to his belief about B's belief. However, belief maintenance among multiple agents in a dynamic environment also interacts with a number of other types of reasoning, especially about the pre-conditions and effects of actions. In addition, default reasoning and persistence (memory) plays a significant role in reasoning about the beliefs of other agents.

As an overview of our approach, we are going to start by defining a belief database, $\mathcal{D}$ (facts that represent beliefs about other agents' beliefs). Then we are going to define an update process that constructs a new set of beliefs $\mathcal{D}^{i+1}$ for a successor state, given $\mathcal{D}^i$. The inputs to the update process will include information the agent has received about current perceptions and events that have just occurred. Furthermore, we will define special types of domain rules, J, that encode various belief justifications for guiding the update process. Thus the agent will operate on a classic sense-decide-act loop. At the start of each cycle i, it will have a set of beliefs $\mathcal{D}^i$. It will then get sense information on its current situation, call it P for perceptions (including actions and messages). Then it will invoke the update procedure to generate a new set of beliefs for the next time step: $\mathcal{D}^{i+1} = Update(\mathcal{D}^i, P, J)$. Finally, it will use its updated model of others' beliefs to make decisions, such as whether it is worthwhile to send new information to a teammate.

3.1 Belief Database

The belief database is a list of tuples that each agent maintains that represent its beliefs about the beliefs of other agents. Each tuple is of the form: $\langle agent, fact, valuation\rangle$. Therefore, a belief database is:

$$\mathcal{D} ::= \{\langle agent, fact, valuation\rangle\}$$

The agents come from a finite set of names of agents A involved in the simulation, and the facts are drawn from a finite set F of facts (propositions). For uniformity, we include the agent's own beliefs in the belief database as well (e.g. $agent = self$).

Typical valuations for propositions would be true and false. In addition, it is important to be able to represent the fact that another agent's belief is unknown, as this will commonly be the case. We represent this explicitly, rather than by absence of information, to avoid any ambiguity between false and unknown. Finally, because of our interpretation of observability, there might be cases where we want to represent that another agent knows whether a given fact is true or false, without being specific as to which. Therefore, we add the valuation type $whether$:

$$valuation \in \{true, false, unknown, whether\}$$

Clearly, their are constraints that inter-relate these valuations. If we use $\nu(\alpha, \phi)$ to be a valuation function that returns the truth value for a given agent α and fact ϕ listed in a belief database, then $\nu(\alpha, \phi) = whether$ can be treated as shorthand for $\nu(\alpha, \phi) = true \vee false$, or equivalently, $\nu(\alpha, \phi) \neq unknown$.

3.2 Perceptions and Actions

It is assumed that an agent will receive a list of perceptions P at each time step. The perceptions will be central to belief updates. By assumption, an agent only has direct access to its own perceptions.

An agent might also come to be aware of actions or events that occurred, which is important for updating beliefs (or at least one's own beliefs about the state of the world). We view actions as standard STRIPS operators, i.e. as discrete-state transitions with pre- and post-conditions (can be modeled by Situation Calculus). However, we do not assume that every agent has knowledge of every action; some actions might be private or remote. Therefore, we only update our model of another agent's beliefs when we have evidence that they are aware of the action.

3.3 Knowledge Base - Belief Justifications

There are a variety of justifications that can be used to update beliefs about other agents' beliefs (as well as one's own beliefs). These can be encoded as a set of domain rules. The types of rules include:

- direct-observation: $(sense\ \phi) \wedge \psi \rightarrow \theta$ - if self senses ϕ and conditions ψ hold, then self should believe θ
- observability: $(obs\ \alpha\ \phi\ \psi)$ - if conditions ψ hold, then agent α will observe (sense) whether ϕ is true or not; note, the truth of ϕ should be evaluated with respect to self's beliefs, which are its own best estimate of the true state of the world
- effects of action: $(effect\ \psi\ \chi\ \theta)$ - if pre-conditions ψ hold, then after the event χ happens, θ will hold; any agent aware of the action will believe the consequences; note that ψ implicitly refers to beliefs in previous state, $\mathcal{D}^i$, while θ refers to $\mathcal{D}^{i+1}$.
- inferences: $(infer\ \phi\ \psi)$ - any agent that believes ψ will infer ϕ
- persistence: $(persist\ \phi)$ - if ϕ was true previously, agents believe it tends to stay true; if it was false, agents believe it tends to stay false
- assumptions: $(default\ \phi)$ - given no other evidence, assume that other agents believe ϕ (as opposed to $v(\phi) = unknown$)

These justifications have different strengths, and it is necessary to take these relative strengths into account when resolving conflicts. There are many situations where conflicting conclusions can drawn from multiple rules. For example, consider reasoning about an agent who (we believe) previously believed that the light was on in a room, and then walks into the room and finds the light to be off. At that moment, we have two rules firing that suggest something about the agent's state of belief: one rule suggests that the agent believes the light is off (by persistence), and another suggests the agent believes the light is on (by observation). Clearly in this case, direct observations by other agents should override their beliefs based on persistence, and this tells us the right conclusion on which to rely.

More generally, the various justification types can be placed in a preference ordering according to strength, and conflicts regarding beliefs about other agents' beliefs can often be seen to occur between justifications at different levels in this hierarchy.

type	*priority*
direct-obs (self)	6
obs (others)	5
action effects	4
inferences	3
persistence	2
defaults	1

The rationale for this particular preference ordering, which is based on analysis of typical domains we are interested in, is as follows. First, direct observation overrides all other justifications, since we assume perceptions cannot be denied. Similarly, observations by other agents override whatever else we might infer they believe. The consequences of actions override inferences because actions often reveal information we did not know. For example, if we thought (inferred) a car was out of gas (e.g. gauge on empty), but we find out someone started it, it must have had gas (assuming we cannot deny the action). Inferences must clearly be given priority over persistence and default assumptions, in order to maintain consistency. In our system, inferences are conjunctive rules that encode constraints of the domain that must hold among various antecedents and consequents, which can provide indirect evidence that the state of something changed and could override the assumption that it did not. Finally, default assumptions have the lowest priority of all and should only be believed if no other information is available about another agent's beliefs. We note that in other domains, different assumptions might apply. For example, if sensors are faulty, then inferences might be more trustworty. However, the preference ordering among the justification types can easily be adapted by re-assigning priority levels.

3.4 Belief Update Algorithm

At this point, we have a database of prior beliefs, $\mathcal{D}^i$, a set of perceptions (and possibly knowledge of a recent action), and a set of rules in the format given above. Now we need to give the algorithm for constructing the new (updated) database of beliefs: $\mathcal{D}^{i+1} = Update(\mathcal{D}^i, P, J)$. Our original idea was to update the beliefs of *self* first, and then use these beliefs to help update beliefs about other agents' beliefs (e.g. to evaluate observability conditions). The beliefs of a given agent would be updated by applying the rules in J in their order of precedence, i.e. perceptions first, then effects of actions, inferences, and so on, down to defaults (last). The second from last step would be to copy over the valuation of persistent facts from the previous database, if they have not otherwise been determined. However, because of the many interactions and dependencies among the beliefs, the order of these updates must be controlled more carefully.

Therefore, we treat this more rigorously as a *prioritized logic program* [Sakama and Inoue, 2000; Brewka and Eiter, 2000], also called an ordered default theory [Grosof, 1995]. We convert each of the belief justifications into a Horn clause,

and label it with a priority level (integer) given in the table above. The semantics of a prioritized logic program is based on a preference ordering over models, where the most preferred models are those in which truth value of each proposition is supported by the strongest rule (with highest priority) in the knowledge base whose antecedents are satisfied. In general, there can be multiple, equally-preferred models (extensions) with inconsistent consequences (divergent conclusions can be drawn, depending on the order in which the rules are applied).

In our initial implementation, we assume that the set of rules (belief justifications) contains no circularities, and hence there is a single stable model. In this case, the dependencies among propositions[2] can be topologically sorted. That is, by placing a directed arc from the consequent of a rule to each of its antecedents, the propositions form a directed acyclic graph (DAG), which can be put into linear order with all edges going in the same direction. Then the truth-values are computed incrementally for each proposition in the sorted order, $Q_1..Q_n$. By construction, the truth-value of the ith proposition Q_i depends at most on the truth-values of $Q_1..Q_{i-1}$, which have already been determined. Determining the truth-value of a given proposition is done by evaluating all the rules that are relevant (i.e. those with proposition Q_i as the consequent) and taking the result of the strongest rule (based on justification type) whose antecedents are satisfied. Note that some rules might also rely on prior beliefs in D or perceptions in P as antecedents, which do not have to be placed in the DAG. For any beliefs with truth value marked as $whether$ at the end of the process, we check whether a specific truth value is believed by $self$, and if so, we automatically promote the other agent's belief to the more determinate value (true or false).

```
BeliefUpdateAlg(D,P,J)
  for each rule C<-A1..An in J,
    create edges (C,Ai)
  topologically sort to get Q1..Qn
  for each Qi
    get all rules R1..Rm in J with head Qi
    evaluate truth of antecedents based
      on D, P, and Q1..Qi-1
    let Rk be strongest rule (by just. type)
      whose antecedents are satisfied
    set truth value of Qi based on Rk
  if v(agt,fact)=whether & v(self,fact)=T
    or F, then set v(agt,fact)=v(self,fact)
  if v(agt,fact) is un-
def, set v(agt,fact)=unk
```

3.5 Integrating Belief Reasoning with Proactive Information Exchange

This belief maintenance algorithm can be used to support more efficient proactive information exchange within teamwork. Reasoning about beliefs of others can be used to filter out redundant information that one agent can infer another agent already knows, e.g. based on observability. We call this enhanced algorithm PIEX for "Proactive Information EXchange." PIEX requires some knowledge of others' goals (which are used to determine relevance), though the tracking of goals may be either fine-grained, where each agent continually broadcasts information on its status, or more coarse-grained, where agents only have approximate information about where each other is in their respective (potentially parallel) parts of the plan, based on generic knowledge of their responsibilities or infrequent synchronization. Note that a proactive message is sent only if it can be inferred that the other agent either does not know (believe) the answer, or believes it incorrectly; if the answer is believed correctly, or if the first agent believes that the second agent knows (believes) whether the information is true, then the message is not sent.

```
PIEX algorithm
  D' <- BeliefUpdateAlg(D,P,J)
  for each agent A and current goal G of A
    for each pre-cond C of G:
      if C is positive literal, let v=false
      else if C is negative lit., let v=true
      if <A,C,v> or <A,C,unknown> is in D'
        Tell(A,C,v)
        update(D',<A,C,v>)
```

4 Experiments

In order to explore the utility of proactive information exchange and the effects of reasoning about observability, we designed a simple synthetic task environment for a pair of cooperating agents. The environment is a simulation based on a variant of the Wumpus World, introduced by Russell and Norvig (1995), which we call *Wumpus Roundup*. In Wumpus Roundup, two agents work together as a team to capture the wumpus. They do this by putting themselves in juxtaposition to the wumpus and then each simultanesously throwing a rope (lasso) on him. However, initially the locations of the ropes and of the wumpus are unknown to the agents. Therefore, the team plan starts with a search for these items.

The physical environment in which this activity takes place consists of a single room with no walls (see Figure 1), divided up into a 10x10 grid (coordinate system). Agents can take one step in any direction at a time, and they know the coordinates of their starting position (the corner of the cave), but they are responsible for keeping track of where they are as they move and where they have been. There are two ropes, one for each agent. They are distinct: agent 1 must obtain rope A and agent 2 must obtain rope B. The initial locations of the ropes and the wumpus in the cave are set randomly. Importantly, in this scenario, the wumpus does not move.

Visual perception is controlled by a sight-radius parameter r; agents can see things up to a distance of r grid cells away from their current coordinate. By setting r to be small, the agents become myopic and must physically visit more grid cells to find things in the cave; by setting r to be larger, they can spot things from a distance, making their job easier.

Some examples of domain knowledge (encoded as justifications[3]) that the agents can use to reason about each other's beliefs include: 1) agents' beliefs in the location of the wumpus tends to persist, 2) an agent will know the location of

[2] Predicates believed by different agents are treated as distinct, but negative and positive literals are merged in the graph.

[3] The actual syntax of rules is not shown due to space constraints.

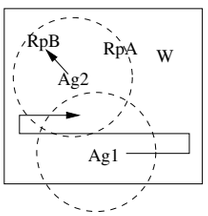

Figure 1: The environment for the Wumpus Roundup world. Ag1 and Ag2 are agents; RpA and RpB and ropes, which must be found; W is the wumpus, to be found and lassoed. Dotted circles indicate visibility radius.

Table 1: Results of Wumpus Roundup experiment with sight-radius r of agents set to 3 or 5, each averaged over 10 random scenarios. PIEX="proactive information exchange."

	without PIEX	with PIEX but no msg filtering	with PIEX and belief reasoning
$r = 3$			
# moves	85.3	59.8	59.8
# msgs sent	0	3.6	1.5
# supressed	0	0	2.1
$r = 5$			
# moves	58.9	45.2	45.2
# msgs sent	0	3.8	1.6
# supressed	0	0	2.2

something if it can see it, and 3) an agent will observe an object if it is in visual range (inferred by a rule based on the sight radius).

In the initial experiment we ran, we set the sight-radius to $r = 3$. We then generated 10 random scenarios (with different locations for ropes and wumpus) and ran the simulation, both with and without proactive information exchange (PIEX). The results are shown in Table 1. The number of 'moves' is the total number of individual steps taken by the agents, summed over both of them. Clearly, the team is more efficient when using proactive information exchange; they capture the wumpus in almost 32% fewer moves (the result is statistically significant by paired T-test at the $p < 0.05$ level). In this case, they can help each other out by sending a message whenever they discover the location of something that the other agent is looking for. For example, if agent-1 finds rope-B, it tells agent-2 (who is looking for it), and this allows agent-2 to interrupt his search-sweep and go directly to the right coordinates. Each agent also tells the other when it discovers the location of the wumpus; two agents searching together is faster than each alone.

With proactive information exchange, there are a total of 3.6 opportunities per run where an agent discovers information potentially useful to the other agent (each eventually sees the other's rope and the wumpus). By maintaining a model of the others' beliefs, the agents can make judicious decisions about whether or not to actually send information based on whether they believe the other agent already knows this information. The 3rd column shows the results with proactive information exchange, but where redundant messages are filtered out by using belief reasoning. Over half of the messages are filtered (2.1 out of 3.6). Yet the performance is the same: 59.8 total moves to capture the wumpus. Similar results were achieved with the visibility radius set to $r = 5$.

5 Discussion and Conclusion

The experiments reported in this paper demonstrate that: a) proactive information exchange can improve team performance in some multi-agent simulations, and b) a large portion of the message traffic can be reduced by intelligently reasoning about beliefs of other agents without introducing a significant drop in performance. A key to tracking other agents' beliefs in a multi-agent system is reasoning about observability. However, this must also be properly integrated with a variety of other justifications for beliefs, such as inferences, persistence, and effects of actions. These justifications have different priorities since some are stronger than and can override others. Therefore, we model the semantics of the knowledge representation as a prioritized logic program, and we give a practical algorithm for deriving strongest conclusions about other agents' beliefs (provided there are no circular dependencies among the propositions).

6 Acknowledgements

This work was supported in part by MURI grant F49620-00-I-3236 from DoD and AFOSR.

References

[Brewka and Eiter, 2000] Brewka, G. and Eiter, T. (2000). Prioritizing default logic. In *Intellectics and Computational Logic - Papers in Honor of Wolfgang Bibel*. Kluwer.

[Finin et al., 1997] Finin, T., Labrou, Y., and Mayfield, J. (1997). KQML as an agent communication language. In Bradshaw, J., editor, *Software Agents*, pages 291–316. MIT Press.

[Grosof, 1995] Grosof, B. (1995). Transforming prioritized defaults and specificity into parallel defaults. In *Proc. 11th Conference on Uncertainty in Artificial Intelligence*, pages 217–228.

[Grosz and Kraus, 1999] Grosz, B. and Kraus, S. (1999). The evolution of shared plans. In Wooldridge, M. and Rao, A., editors, *Foudations of Rational Agency*, pages 227–262. Kluwer.

[Isozaki and Katsuno, 1996] Isozaki, H. and Katsuno, H. (1996). A semantic characterization of an algorithm for estimating others' beliefs from observation. In *Proc. AAAI*, pages 543–549.

[Sakama and Inoue, 2000] Sakama, C. and Inoue, K. (2000). Prioritized logic programming and application to commonsense reasoning. *Artificial Intelligence*, 123:185–222.

[Tambe, 1997] Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124.

[Wooldridge, 2000] Wooldridge, M. (2000). Reasoning about visibility, perception and knowledge. In Jennings, N. and Lesprance, Y., editors, *Intelligent Agents VI, ATAL'99*. Springer Verlag.