Towards Semantically Coherent Collaboration

Hadi Sabaa
Department of Computer Engineering and Computer Science
University of Arkansas
Fayetteville, AR 72701, USA.
hsabaa@uark.edu

Henry Hexmoor
Department of Computer Engineering and Computer Science
University of Arkansas
Fayetteville, AR 72701, USA.
hexmoor@uark.edu

Abstract

Ontologies are fundamental for communication-based collaboration. Common ontologies enable agents to successfully communicate. Agents in a multi-agent system might share the same ontology or might have different ontologies. In this paper we present a methodology that enables agents characterized by minimally different, lower ontologies to meaningfully communicate. Our approach is based on mediating ontologies between the communicating agents.

1. Introduction

Ontology technologies aim to capture and facilitate interoperability of common concepts within an interest community [1]. Part of this effort is to construct the most abstract, shared concepts often called formal- or upper-ontology [2]. Another part of this effort captures lower- or domain- ontologies. Yet other, more pragmatics efforts focus on ease of generation and maintenance of ontologies [3].

In a multi-agent system where agents have different ontologies, communication-based collaboration is cumbersome due to lack of shared understanding [4]. The problem arises because it is often the case that the agents communicating were originally developed for different purposes [5]. Hence, a need for developing a framework for meaningful collaboration is vital in such cases.

Clearly, an approach based on developing a *common ontology* and requiring all communicating agents to adopt that ontology would solve the problem at hand [4]. However, we shall rebuff such an approach, for it is not only impossible to implement, but worse still, even developers of taxonomies often disagree on terminology [4], [5].

We are not interested herein in discussing the aspects of coordination, role assignment, performance, language understanding, and lexical issues relating to the language used in communication. In this paper, our sole purpose will be to develop a methodology that provides agents characterized by a common upper ontology yet having minimally different lower ontologies, the ability to meaningfully collaborate.

Mediation is a promising tool for providing agents with such a methodology [6]. During the mediation process, as described in further detail in section 2, a mediator is created dynamically, which is computationally inexpensive [7]. Hence, Mediation is dynamic in a sense that it takes place as communication proceeds [8]. Another reason for adopting mediation is its applicability to agents collaborating in any domain [9]. Unlike the common ontology approach where a specific common ontology needs to be developed prior to collaboration, mediation does not require such a priori ontology. The mentioned benefit is very important whenever newly arriving agents having a different ontology join the group [10]. Hence, mediation lends itself to communication-based collaboration in a distributed multi-agent system [11], [12]. Also, mediation, as described in section 2, utilizes idle agents and so increases efficiency.

Each agent has a unique *Identification Number* (*ID*). When an agent sends a message, she includes her *ID* as part of the message.

An agent can have one of two roles, either *active* or *idle*. An *active* agent is currently performing a task. An

idle agent is currently not *active*. An *idle* agent becomes *active* when she is assigned a task. An *active* agent becomes *idle* when she accomplishes her task.

An agent can have one of two statuses, either *ordinary* or *mediator*. An *ordinary* agent performs regular tasks. A *mediator's* sole task is to mediate different ontologies between two *ordinary* agents. An agent's default status is *ordinary*. If an *ordinary* agent receives a request from another *ordinary* agent to play the role of a *mediator* and agreed to play that role, it becomes *a mediator*. Upon completing its task, a *mediator* will become *idle* and *ordinary*. An *ordinary* agent can be *active* or *idle*. A *mediator* is always *active*.

In our approach, confidentiality is critical for the agents to perform their tasks. Thus, peer level agents have ontologies that are impermeable to other agents. Hence, peer level agents do not have the ability to *self mediate*. Only when an agent becomes a mediator, will she have the ability to access other agents' ontologies. Also, we assume a methodology for assigning the role of a mediator to an agent when needed. Agents are able to identify other agents in their proximity as well as their current role and status.

For the mediation process to effectively take place, each concept in an agent's ontology should have an associated list of *functional specifications.* The latter is a form of delineation that uniquely describes a particular concept by identifying its primary functions. Of course, all the concepts in a *functional specifications list* are found in the corresponding concept's ontology. No two concepts in a particular ontology have the same list of functional specifications. However, concepts can have functional specifications lists overlapping with those of other concepts in the same or different ontology.

A functional specification list is comprised of words found in their root form. For accuracy of mediation, we require a a *functional specifications list* to include at least five functional specifications. We assume that agents in our system have ontologies as that described.

2. An algorithm for a two-party mediation

We will present an algorithm for mediating ontologies between two parties.

Let us denote an agent sending a message as *S*, an agent receiving a message as R, the message in transit as *m*, a mediator agent as *Med*, and the commitment level as *CL*. *CL*, which can be set to different values, is the level to which agents comply with other agents' directives.

1. Sender agent sends a message denoted m to receiver agent.
2. If receiver does not understand some constituent(s) of m
   {
3. Receiver agent deploys a mediator agent and provides the latter with misunderstood m.
4. Mediator discovers the ontological origin of m in the sender's ontology.
5. Mediator explores the corresponding, related region in the receiver's ontology.
6. If a match is found
7. Notify receiver of the matching region, else
8. Mediator recommends a modification to the receiver's ontology.
9. Depending on the degree of collaboration, the receiver enriches its ontology.
   }

Figure 1. An algorithm for mediating ontologies between two parties.

Figure 1 outlines our proposed algorithm. We assume that all agents in the system uniformly adhere to a *CL*. If an agent *R* receives a message *m* from a sender agent *S* and does not understand constituent(s) of *m*, owing to differing lower ontologies between *S* and *R*, *R* would search for an idle agent and would send the latter the misunderstood constituent(s) of m along with both *S*'s and R's *IDs*. The *ID's* are used by the mediator to identify the agents involved in the message. Depending on the current value of *CL*, the idle agent might accept the mediator role or reject it. If the idle agent refused to play the role of a mediator or in absence of idle agents, *R* will create a *Med* to play the role of a mediator. The newly created *Med* has a duplicate of *R*'s ontology; however, she has a different *ID*.

Since we assume that the same prepositions and connectives are innate in all agents' ontologies, an agent does not include them as part of the misunderstood constituents of *m*. Upon receiving the misunderstood constituent(s) of *m*, *Med* will discover the ontological origin of the appropriate concepts in *S*'s ontology and examine their corresponding *functional specifications list*.

By inspecting and comparing this list with those found in *R*'s ontology, *Med* will explore the latter for a corresponding match for the list found in *S*'s ontology. Two lists are considered to match if there is a portion of the first that is found in the other. If more than one match is found in *R*'s ontology, the list with more in common with the list found in *S*'s ontology is chosen to be the match. If a match is found, *Med* will inform *R* of such a match. Else, *Med* will propose an adjustment to R's ontology. Here too, depending on the current value of *CL*, the receiver agent will either attempt to comply with *Med*'s recommendation or otherwise reject it.

In our algorithm we assume that if a concept is found in two different ontologies, then the corresponding concepts mean the same thing to their respective agents.

A Mediation Example:

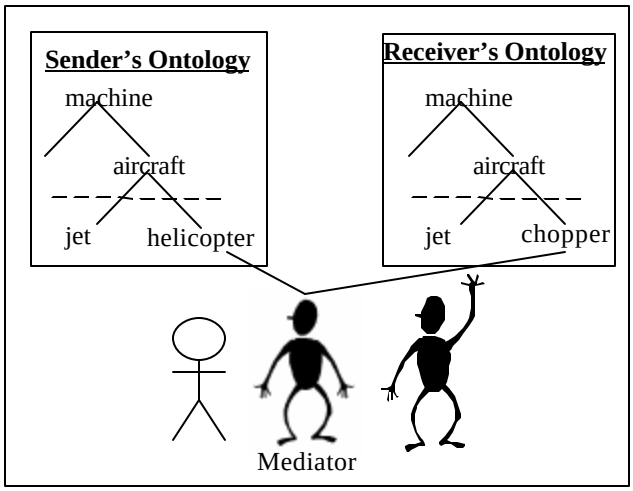

Figure 2. An illustration of the process of mediation between two parties.

Figure 2 is a pictorial representation of the mediation process. The upper ontologies include the concepts of *machine* and *aircraft.* The lower ontologies comprise of the concepts of *jet, helicopter, and chopper*. The dotted line separates lower and upper ontologies. As shown in figure 2, both the sender and the receiver share a common upper ontology. However, some parts of their lower ontologies differ. Both the sender and the receiver are familiar with the concept of a *helicopter*/*chopper* however; they use synonyms to denote that concept.

Let's suppose that the sender agent sent a message containing the concept of a *helicopter* to the receiver agent. The latter will not, of course, grasp that concept for it is not part of its own ontology. Assuming that the *CL* is set to the highest value and that an idle agent exists, the receiver will send the concept of a *helicopter* to the idle agent who will play the role of the mediator. Figures 3 and 4 sketch the *functional specification lists* corresponding to the concepts of a *helicopter* and that of a *chopper* respectively.

- Is an aircraft
- Uses engine
- Utilizes blades
- hovers
- needs fuel

Figure 3. Functional Specifications list of a *helicopter*

- Is an aircraft
- Uses engine
- Utilizes blades
- Is a machine
- needs fuel

Figure 4. Functional Specification list of a *chopper*

The mediator will attempt to match the two concepts by exploring the receiver's ontology to find a counterpart for the *helicopter*'s *functional specification's list*. In this case, the concept of a *chopper* has an associated *functional specifications list* that shares most of those of a *helicopter*. Hence the mediator finds a corresponding concept in the receiver's ontology for the concept of a *helicopter* in the sender's ontology, which is a *chopper* in this case.

Notice that a concept in an agent's ontology might share a considerable part of its functional specifications list with other concepts for various reasons. One reason could be that a concept is a subclass of another. In our example, a *helicopter* is a machine; however, a machine is not necessarily a *helicopter*. This means that a machine's *functional specifications list* will have a lot in common with that of a *helicopter*. Nonetheless, this should not be a problem for the mediator while mediating for it will search for a concept associated with a *functional specifications list* that most closely corresponds to that of the concept she is trying to mediate. So even though a machine would be a possible match for a *helicopter*, since a *chopper*'s *functional specifications list* shares more with that of a

helicopter than the machine's *functional specifications list*, the mediator will choose it as the *best match.*

Another reason for two concepts to have relatively similar *functional specifications lists* is due to the fact that they are both subclasses of the same super class. For example, both a *chopper* and a jet might have a *functional specifications list* entry "it flies". However, for the same reasoning mentioned above, this, again, should not be a problem for the mediator while mediating ontologies.

3. n-party mediation

Our proposed algorithm, described in section 2, scales up to any number of agents. Next, we describe our algorithm in general terms.

Let there be n agents in a given environment. The need for n-party mediation arises when an agent broadcasts a message to multiple agents having different lower ontologies than that of the sender.

Agents sharing the same ontology will not understand the same constituent(s) of the broadcasted message [13] Say there exists m such agents denoted as group A. Then, according to the algorithm shown in Figure 1, m mediators will be deployed to simultaneously mediate between the sender's ontology and that of group A regarding the same message. Notice that more than one group of agents where members of the same group share the same ontology might exist. Therefore, it is inefficient to deploy one mediator for each member of such a group. Rather, it is more efficient to deploy one mediator for a whole group of agents sharing the same ontology.

Figure 5 shows an algorithm for mediating ontologies between n parties. Agents know which other agents share the same ontology as theirs and which do not. Upon misunderstanding constituent(s) of a broadcasted message owing to differing lower ontologies between the sender and the receivers, agents sharing the same ontology deploy an agent to play the role of the mediator and send the latter the misunderstood constituent(s) of m, each of the agents' *ID*, as well as the sender's *ID*. The mediator will mediate between the sender and any one agent in that group as in two-party mediation and then broadcast the result to all the members of the group.

We assume a methodology for agents to identify all other agents having the same ontology. We further assume a methodology for a group of agents sharing the same ontology to agree on deploying one agent to play the role of a mediator upon receiving a broadcasted message.

At any point in time, there might exist any number of groups of agents where members of the same group share the same ontology.

1. Sender agent broadcasts a message denoted m to all agents.
2. If receiver does not understand some constituent(s) of m
   {
3. Agents sharing the same ontology form a group and deploy one mediator agent and provide the latter with misunderstood m, each of the agents' *ID*, as well as the sender's *ID*.
4. Mediator discovers the ontological origin of m in the sender's ontology.
5. Mediator explores the corresponding, related region in any of the group members' ontology.
6. If a match is found
   send result to all members of the group, else
   Mediator recommends a modification to the group's ontology.
7. Depending on the degree of collaboration, the receiver rectifies its ontology.
   }

Figure 5. An algorithm for mediating ontologies between n parties

Figure 6 is a pictorial representation of a n-party mediation. Agents in group A share a common ontology. Similarly, agents in group B share a common ontology. Group A, group B, and the sender agent share a common upper ontology. However, parts of their lower ontologies differ.

The sender agent broadcasts a message denoted m to all the agents in the system. Owing to different, lower ontologies, members of group A and those of group B will not understand constituent(s) of m. Since all the members of group A share the same ontology, they will not understand the same constituent(s) of m. Similarly, because members of group B share the same ontology, they will not understand the same constituent(s) of m. However, group A's misunderstood constituent(s) differ from those of group Bs misunderstood constituent(s) since both groups share different lower ontologies.

Group A deploys an agent to play the role of a mediator and sends the latter the misunderstood constituent(s) of m, all the group members' *IDs*, as well as the sender agent's *ID*. Group A's mediator identifies the sender and any of Group A's members using their respective *ID's* and mediates between them as in a two-party mediation. Subsequently, the mediator broadcasts the results to all the members of group A identifying them by their distinctive *ID's*.

Similarly, group B deploys their own mediator who will mediate between the sender agent and any of group B's members. Afterwards, the mediator broadcasts the results to all the members in group B.

Hence, rather than having a mediator for each agent who misunderstood constituent(s) of a broadcasted message, the number of mediators required for a single broadcasted message is reduced to the number of groups having a unique ontology. In a multi-agent system where thousands of agents exist, the decrease in the number of mediators enhances performance and efficiency dramatically.

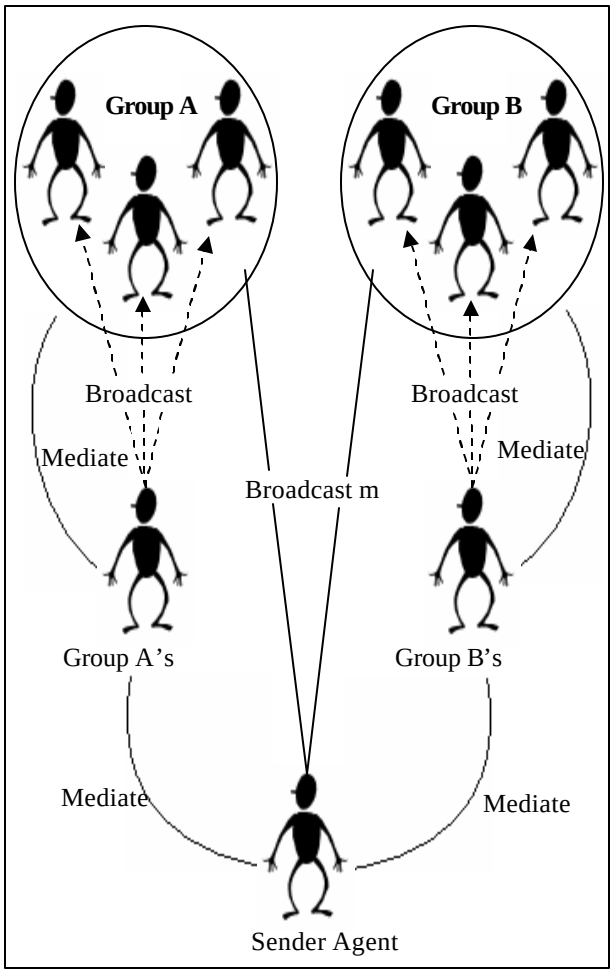

Figure 6. . An illustration of the process of mediation between n parties

Discussion:

The algorithm shown in figure 1 has a running time $O(k_1.k_2)$ where k_1 is the number of concepts constituting the sender's ontology and k_2 is the number of concepts constituting receiver's ontology.

The algorithm shown in figure 2 has a running time $O(n_1.n_2.p)$, where n_1 is the number of concepts constituting the ontology of one group of agents whose members share the same ontology, n_2 is the number of concepts constituting the sender's ontology, and p is the number of groups where each group has a unique ontology shared by all of its members. Note that a group can have any number of agents and that the sender might belong to a group of agents sharing the same ontology.

4. Conclusions

A framework for providing agents characterized by a common upper ontology however, having minimally different, lower ontologies the ability to meaningfully collaborate is presented. The approach adopted is based on mediating ontologies by deploying an agent to play the role of the mediator. Our approach is dynamic in a sense that it takes place as communication proceeds and hence does not require a common ontology to be developed prior to collaboration. Thus, our approach can accommodate dynamic arrival of agents and permits their collaboration with currently existing agents regardless of their respective ontologies. When mediation is between more than two parties, the number of mediators required for a single message is equal to the number of groups where each group is characterized by a unique ontology. We plan to explore application of ontology-based mediation in various, simulated agent communities for validation and further explorations.

References

[1] P. Grenon, " Knowledge management from the Ontological Standpoint," in *Proceedings of the Workshop on Philosophy and Knowledge Management at WM2003*, 2003.

[2] M. Uschold and M. Gruninger, "Ontologies: principles, methods and applications," in *The Knowledge Engineering Review*, 1996, pp 93-136.

[3] J. H. Gennari, W. Grosso, and M. Musen, "A Method-Description Language: An initial ontology with examples," Stanford Medical Informatics, Stanford University, Tech. Rep. SMI-97-0695, 1997.
[4] A.E. Campbell and S.C. Shapiro, "Algorithms for Ontological Mediation," Dep. of CS and Engineering, State Univ. of New York at Buffalo, Tech. Rep. 98-03, 1998.
[5] T.R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Knowledge Systems laboratory, Stanford University, Tech. Rep. 93-04, 1993.
[6] D. McGuiness, "Ontologies for Information Fusion," in *Proceedings of the Sixth International Conference on Information Fusion*, 2003, pp. 650-657.
[7] N.F. Noy and M.A. Musen. "SMART: Automated support for ontology merging and alignment," in *Proceedings of the twelfth Banff Workshop on Knowledge Acquisition, Modeling, and Management*, Banff, Alberta, Canada, 1999.
[8] T.R. Gruber, "Ontolingua: A mechanism to support portable ontologies," Knowledge Systems Laboratory, Stanford University, Tech. Rep. KSL-91-66, 1992
[9] T.R. Gruber, "A translation approach to portable ontology specifications," Knowledge Systems Laboratory, Stanford University, Tech. Rep. KSL 92-71, 1993.
[10] D. L. McGuiness, R. Fikes, J. Rice, and S. Wider, "An Environment for Merging and Testing Large Ontologies," in *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, 2000.
[11] W. Swartout, R. Patil, K. Knight, and T. Russ, "Toward distributed use of large-scale ontologies," Information Sciences Institute, University of Southern California, 1996.
[12] M. Uschold and M. King, "Towards a methodology for building ontologies," University of Edinburgh, Tech. Rep., July 1995.
[13] N. Noy, M. Musen, "An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support," in *Proceedings of the Workshop on Ontology Management at Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999.