Reconciling Ontologies for Coordination among E-business Agents

Jingshan Huang, Jiangbo Dang, and Michael N. Huhns

Computer Science & Engineering Dept., University of South Carolina, Columbia, SC 29208, USA
{huang27, dangj, huhns}@engr.sc.edu

1. Introduction

To meet the demands of dynamic and open e-markets requires e-businesses to coordinate their activities. The first step towards this coordination is for e-business agents to understand each other's service description. Using ontologies can aid in this understanding. However, independently designed ontologies usually have heterogeneous semantics. We present a schema-based approach to reconcile ontologies and then introduce compatibility vectors to solve the problem of how to select more compatible agents with which to interact. For details, see http://www.cse.sc.edu/~huang27/paper/AAMAS%2006.pdf.

2. Ontology Heterogeneity and Our Solution

An example scenario of the interaction within an e-business environment can be envisioned as follows:

1. A number of agents form an e-business community (EBC) within which services from different agents might be integrated to render a better service.
2. The agents outside this EBC can request help from the community and make use of its services, either the original ones or the integrated one.

Because there is no ontology which is global and accepted by every agent, ontological heterogeneity among agents becomes an inherent characteristic in an EBC. Therefore, two major problems are envisioned here. First, during the formation of an EBC, how can it be ensured that all agents within the community have no problem in understanding each other's ontology? Second, an agent seeking coordination from outside this community would like to choose those agents that understand its ontology best. How can it ensure this selection is a correct one?

In order to solve the first problem, we need an approach to match/align ontologies from different agents. To tackle the second problem, we introduce compatibility vectors as a means of measuring and maintaining ontology quality. By determining the compatibility for each constituent agent along with the formation of an EBC, not only the agents outside this community are able to select the best agent(s) with ease, but also a better mutual understanding of ontologies within the EBC is obtained.

3. A Schema-based Ontology Merging Algorithm

Our goal is to develop a methodology for constructing a merged ontology from two original ones. The methodology can then be applied iteratively to merge all ontologies within an EBC.

The ontology merging is carried out at the schema level. Internally we represent an ontology using a directed acyclic graph $G\ (V, E)$, where V is a set of ontology concepts (nodes), and E is a set of edges between two concepts, i.e., $E = \{(u, v) \mid u$ and v belong to V and u is a superclass of $v\}$. In order to merge two ontologies, G_1 and G_2, we try to relocate each concept (node) from one ontology into the other. We adopt a breadth-first order to traverse G_1 and pick up a concept C as the target to be relocated into G_2. Consequently, at least one member of C's parent set Parent(C) in the original graph G_1 has already been put into the suitable place in the destination graph G_2 before the relocation of C itself. The time complexity of this algorithm is $O(n^2)$, with n the number of concepts in the merged ontology.

4. Ontology Distance and Compatibility Vectors

Along with the formation of an EBC, we create a *center* ontology by merging all the original ontologies; then the distances from the latter to the center are suitably encoded in the compatibility vectors, and can be adjusted efficiently and dynamically. Based on the information contained in the vectors, agents are supposed to understand the ontology from each other without trouble. In addition, the agent from outside this community will have no difficulty choosing the agents with good compatibilities.

4.1. Concept Distance and Ontology Distance

The concept distance in original ontologies is represented by the amount of information missing, i.e., the number of relationships known in the center but not in the original ontology. The following equation formalizes the concept distance: $d_{concept} = w_1 * n_{sub\text{-}super} + w_2 * n_{other}$, with the constraint of $(w_1 + w_2 = 1)$. $n_{sub\text{-}super}$ is the number of sub/superclass relationships missing in the original ontology, and n_{other} is that for other relationships. w_i is the weight assigned to different kinds of relationship.

After each concept distance has been calculated, the ontology distance between the original ontology and the center can be figured out as: $d_{ontology} = \sum_{i=1}^{n} w_i * d_{concepti}$, where $d_{concepti}$ is the distance between a pair of concepts, n is the number of concepts in the center, and w_i encodes the importance of a concept in its ontology. We use the percentage of the number of relationships to represent this measurement. For example, if ontology_1 has 100 relationships in total, and concept "Spatial" has 15 relationships, then the weight for this concept in ontology_1 is 0.15.

4.2. Compatibility Vectors

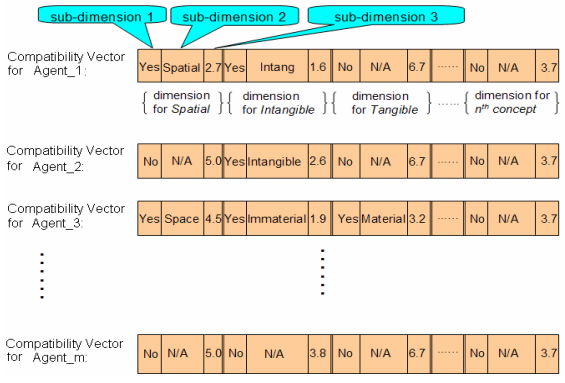

Figure 1. Compatibility Vectors

Inside the center, there is a set of compatibility vectors, one for each original ontology. A compatibility vector consists of a set of

dimensions, each corresponding to one concept in the center. Each dimension has three sub-dimensions. The first one tells us whether the original ontology understands this concept or not; the second one records the concept name in the original ontology if the latter does recognize that concept; the third one encodes the corresponding concept distance. An example of compatibility vectors is shown in Figure 1.

For the first concept ("Spatial") in the center ontology, provider_1 knows it as "Spatial" and has a concept distance of 2.7; provider_3 also understands this concept, but with a different name ("Space") and a bigger concept distance of 4.5; neither provider_2 nor provider_m recognizes concept "Spatial", therefore, they have the same concept distance (5.0).

4.3. Dynamically Adjusting Compatibility Vectors

When there is only one agent, its compatibility is perfect. In the compatibility vectors stored in the center, each concept distance has a value of zero. However, with the adding of new agents into this EBC, the compatibilities for existing agents might be changed.

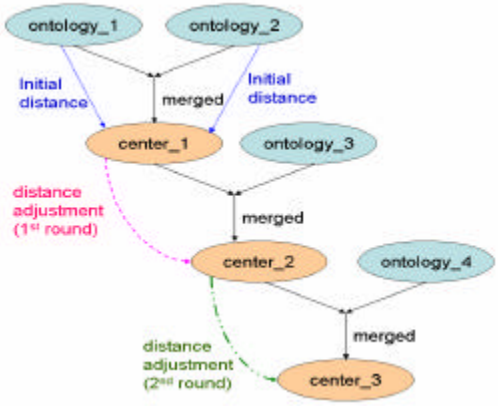

Figure 2. Dynamic Adjustment of Compatibility Vectors

An example is shown in Figure 2. After ontology_1 and ontology_2 are merged to generate center_1, the distance between these two and the merged center_1 is calculated and stored in the compatibility vectors of center_1. Upon the joining of ontology_3 and the generation of center_2, the compatibility vector for center_1 in center_2 is calculated and integrated with the compatibility vectors for ontology_1 and ontology_2 in center_1; then we generate the compatibility vectors for ontology_1 and ontology_2 in center_2. We have an algorithm (whose details are omitted here) to accomplish the above adjustment, and its time complexity is $O(n * log\ n)$, with n the number of concepts in the center. Figure 3 exemplifies our algorithm.

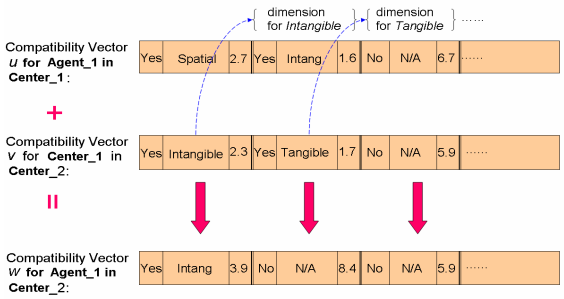

Figure 3. Example of Vector Adjustment

4.4. Ontology Understanding via Compatibility Vectors

The center maintains the compatibility vectors for all original ontologies. If two agents would like to comprehend each other's ontology, they can refer to the corresponding compatibility vectors and obtain enough information in mutual understanding. In addition, when an agent from outside this EBC requests for agent(s) to coordinate with, it would like to choose those that understand its ontology best. The requesting agent first compares its own ontology with the center, and then searches in the compatibility vectors to find all agents understanding the concept of its interest. If there is more than one candidate, the coordination request will be sent to those with good compatibilities, that is, with small concept and/or ontology distance.

4.5. Correctness and Complexity of Compatibility Vectors

To record and maintain the proper compatibility of each agent inside an EBC, the key is to obtain a correct center ontology by which to evaluate the distance from it to each original ontology, and thereby acquire the corresponding compatibility vector. When a new agent joins the EBC, instead of communicating with each existing agent, it only talks with the center ontology. Therefore, if we can prove that the newly merged ontology is a correct new center, the correctness of compatibility vectors is guaranteed.

Lemma 1. When we merge two ontologies A and B using the algorithm in Section 3, the result is the same regardless of whether we merge A into B or merge B into A (proof omitted).

Theorem 1. The final result of merging a number of ontologies is identical no matter by which order the original ontologies are merged using the algorithm in Section 3 (proof omitted).

The time complexity of establishing an EBC, along with the achievement of mutual understanding of ontological concepts, is on the order of $O(n^2 * m)$, with n the number of the concepts in the center, and m the number of original ontologies. For the ontology merging, $O(n^2 * m)$ is needed, because we need to merge m ontologies, and each merging procedure takes time $O(n^2)$. In addition, in order to dynamically update the compatibility vectors, extra time will be spent. $O(n * log\ n)$ is needed for updating one agent, so the extra time for all agents is $O(n * log\ n * m)$. Therefore, the total time complexity becomes $O((n^2 + n * log\ n) * m)$, which is on the same order of $O(n^2 * m)$. For agent selection, the time complexity is $O(n^2)$. We only need to compare the ontology from the requesting agent with the center ontology.

5. Experiment Results

A set of experiments have been conducted to show the correctness and efficiency of our approach. Details can be found in http://www.cse.sc.edu/~huang27/paper/AAMAS%2006.pdf.

6. Future Work

Some future work is envisioned here: (1) our current approach makes use of a center ontology, but introduces the problem of how to handle the vulnerability issue inherent in this centralized solution, (2) how to maintain compatibility vectors when existing agents modify their corresponding ontologies, and (3) what kind of mechanism is suitable if we simultaneously consider qualities of both ontologies and services.