Abstract

This paper introduces a multi-agent system (MAS) architecture and a relevant algorithm for multiple agents to manage and deploy ontologies in a dynamic environment. Firstly, an ontology description is given followed by a three level conceptual model which shows the background of our work. Then a MAS architecture is presented. Finally, we demonstrate how this approach enables ontology evolution and integration by using the JADE platform.

1. Introduction

Ontologies [3], as explicit specifications of conceptualisation, facilitate interoperability between heterogeneous systems. Ontology-based approaches [4, 5] have shown advantages both at design and run time phases for information integration and system interoperability.

Unfortunately, it is hardly conceivable that a single ontology can be constructed and applied for all involved parties and applications. Moreover, a changeable environment enforces underlying ontologies evolving over time. Therefore, finding mappings among multiple ontologies turns out to be a practicable and a reconciliation approach.

Ontology mapping faces new challenges in the context of semantic Web, especially concerning higher levels of heterogeneity, evolution, distribution, autonomy and limitations on the representation technology. Manually maintaining ontology is error-prone. Only until recently have multi-agent systems (MAS) been used in ontology management which direct not only how to use ontology but also how to manage ontology. The MAS approach is believed a natural and intuitive way in handling dynamics. Contributions of this paper are: an ontology description is presented to speed up ontology mapping. A new MAS architecture and a relevant algorithm are proposed to facilitate ontology automatic mapping to some extent.

Our discussion is focused on a VO (virtual organisation) scenario where two parties from two education institutes investigate possible cooperation for the sake of mutual benefits. We do not distinguish terms between parties, partners, participants and agents in this paper; herein we mean they are software agents. Besides this, we use “conceptual model” and “ontology” alternatively to indicate the conceptual schema of a domain. In this paper, we also suppose that all data are mapped onto the same representation level to conform to a normalised uniform representation.

This paper is organised as follows. Section 2 discusses the ontology mapping problem. Section 3 proposes our basic ontology architecture and MAS structure based on this basic architecture. We focus on ontology mapping between two ontologies by presenting an algorithm based on the MAS architecture in Section 4. We demonstrate our approach in Section 5 with an example to show our preliminary work. Section 6 discusses the related work, and finally, Section 7 is the conclusions and future work.

2. Ontology mapping problem

In the context of ontology mapping, there are some assumptions as follows:

(1) Those multiple ontologies stem from a same domain and own same primitive elements, we may call them the baseline of the conceptual models.

(2) Mapping is a process of explaining the source instances of the source conceptual model in the target conceptual model based on primitive concepts.

(3) The main focus is to find mapping functions from the source ontology to the target ontology for specific concepts. Therefore, these mapping functions will be demonstrated in a form of rules in CLIPS, Prolog or even logic expressions.

(4) We only discuss that for each concept in the source ontology, there must have one concept associated with it in the target ontology. That is, the mapping is one-to-one. For other cases such as no corresponding entities in the target ontology or in the target ontology that the same concept is defined more than once, please refer to the future work.

(5) Since taxonomies are core components of ontologies, we can borrow the object orientation concept, that is, a subclass inherits all properties of its super class.
In this paper, we follow the definition in [2] in which the taxonomy tree is a typical organisation to describe an ontology. We define an ontology which specifies a domain mode, $T$, in terms of concepts, attributes, and relations, a tuple in the form of $O := (C, R, L, E)$, where $C$ stands for a set of concepts including abstract concepts and primitive ones where abstract concepts include sub-concepts, while primitive ones are the baseline of people’s knowledge of that domain; $R$ defines binary relationships such as is-a, property-of, and instance-of; $L$ is a set of logic operators such as $\land$, $\lor$ and $\neg$; $E$ can be $=$, $\subseteq$ to specify relations between concepts. DeMorgan’s laws are used to deduct expressions to only including conjunction and negation operators. We may simply note ontology $O$ as $O = \bigcap_{j=1}^{n} \bigcap_{i=1}^{m} R_{i,j} \cdot C_{j}$ under model $T$. Mapping may be an exact match, partial match or mismatch.

With each element, $C_{i}$, in set $C$, ($C_{i}$, $C_{j}$ are used to represent concepts from source ontology and target ontology respectively), we define $C_{i} = \bigcap_{j=1}^{n} R_{i,j} \cdot C_{j}$, where “:*=” might be one of $=$ and $\subseteq$ according to $T$. This formula implies that each concepts might be decomposed into sub-concepts until primitive concepts in forms of $C_{i}$ associated with relations $R_{i,j}$. In that way, all those sub-concepts are combined by operator “$\cap$” corresponding to the logic expressions in the conceptual model under a specific domain theory.

Clearly, now the mapping problem needs to consider the following issues:

1. How can we find such a mapping if there exists a mapping between $C_{i}$ and $C_{j}$, $i, k \in N$ (natural number)?

2. How can we explain such a mapping in terms of another ontology (the target ontology)?

3. How can the mapping benefit the Knowledge Base (KB) and in turn speed up ontology integration?

The following sections will address the above three issues with the focus on the first two.

3. Conceptual architecture

A three layered architecture is presented here. At the bottom common knowledge layer, we assume all participants have common knowledge (primitive concepts) about the domain. It is the baseline of people’s knowledge. On top of the common knowledge layer, it is the representation layer. Though the representation languages such as RDFs, DAML+OIL, and OWL differ in their terminologies and expressiveness, the ontologies that they model essentially share the same features to some extent. We suppose that all data are mapped onto the same representation to conform to a normalised uniform representation. Once the syntactic expression is cleared away, the ontology mapping problem becomes much clearer. It directly benefits semantic mapping in ontology. Finally, the top ontology semantics layer is a tuple in the form of $O = (C, R, L, E)$. That is, concepts may follow the same way to extend to primitive concepts, which we call common knowledge in the paper.

The ontology-based application architecture is shown in Figure 1. Each party in the scenario includes three parts. Ontology-based applications (App1 and App2) are one part while specific ontologies (Onto1 and Onto2) and relevant KBs (KB1 and KB2) are another part. An agent system as well as the mapping module and the refining module consist of the third part. Due to the space limit, we only focus on the mapping module in this paper.

![Figure 1. Ontology-based application architecture](image)

Each agent in Figure 1 has its own structure as shown in Figure 2. Agents are likely to offer functionalities such as learning and rule generation to refine ontologies and map between multiple ontologies. The results of learning and inferencing are expressed with rules which may be in forms of CLIPS, Prolog and even logic expressions. The communication between agents hosted on a same computer is conducted by ACLMessage2. Protocols such as HTTP or SOAP can be used in geographically distributed environments.

![Figure 2. Single agent architecture](image)

4. Multiple ontology mapping with MAS

Ontology mapping associates source ontology nodes with target ontology nodes, including transforming source ontology instances into target ontology instances based on conceptual models. We use subgraphs to represent specific concepts. For simplicity, each node (i.e. a concept in an ontology) in the graph is notated by a label like $C_{i}$ instead

---

1 OWL (http://www.w3c.org/2004/OWL) is the official successor of all of available ontology languages.

2 Jade platform (http://jade.tilab.com/) is used in the implementation.
of a real name and a real number varies from 0 to 1. Numbers between [0,1] are used to indicate to which extent these two concepts coming from the source ontology and the target ontology would be thought the same or matching one another (here number 1 means exactly match, while 0 means completely mismatch). Any number between (0,1) stands for a partial match: a bigger number means a better match. We expect exact matching as much as possible, but obviously, it is not always the case. So in most cases, we can only achieve partial matches. When a match is completed, a path from the root to that particular node is highlighted to indicate its location (also relevant relations with its upper nodes) within the ontology in a form of, say, StudentStaffResearchStudent. PhDStudent (given “PhDStudent” is specified here). Apart from the clear indication of relations as mentioned above, it has significant effects in the context of semantic Web because it then provides namespaces to avoid name conflicts between two ontologies. Of course, some predefined rules are still needed to solve conflicts and assign higher levels to some properties to depict certain concepts naturally and intuitively as detailed next.

For clarity, we would like to summarise assumptions before we start to detail the mapping process. We only consider ontologies which satisfy the consistency, completeness and integrity. For other cases, please refer to the future work. The meta-level schemas are known by embedded agents. Therefore, agents would like to exhibit most likely candidates (nodes) in their minds when a new query is presented. That is to say, mapping always exists but a mechanism must be provided to choose which mapping is better in the context of semantics. The real number attached to the node in the subgraph is such a kind of criterion: the bigger the better in our case.

Figure 3 depicts two subgraphs excerpted from running examples respectively by ignoring nodes’ real meanings. Initially, we assume that each node is assigned 1 by default (default number 1 attached to each node is omitted). Here number 1 means it exactly matches in terms of itself. The number will change during the mapping process.

![Figure 3. Subgraphs with relations explicitly represented](image)

Generally, ontology mapping goes on with the following steps (RA and SA stand for receiver and sender agents):

1. RA extracts key words from the query; RA may contact SA for more information regarding some key words.
2. RA provides suitable candidates presented in a form of subgraphs. Every candidate appears with a subgraph to indicate relationships with its direct upper node and direct successive nodes. In the end, the situation looks like this: there is one source subgraph (S_i) along with several target subgraphs (S_j, j ∈ [1..n]) which are waiting to sort out according to some criteria stated in step (3).
3. Since we use inheritance concept to describe the “is-a” relation in the ontologies, the source concept C_i = ∩_{j=1} C_{jA} can be described with the “property-of” relation in the end. Suppose they are annotated by A_i, j ∈ [1..n] (it is requested that A_i be decomposed into primitive concepts), we define that A_i = {A_j | j ∈ [1..n]} . We use the same annotations for the target concepts. That is: C_i = ∩_{j=1} C_{jA} , and in the end annotated by A_i , j ∈ [1..m], and A_i = {A_k | k ∈ [1..m]}.
4. Suppose cardinalities of A_i and A_i are in forms of | A_i | and | A_i | respectively, for example, | A_i | implies that how many attributes are involved to define a specific concept. In the system, we compare their individual attributes and decide to what extent these concepts match one another. That is, we first compute intersection A_i ∩ A_i . Four cases may occur:
   (i) A_i ∩ A_i = ∅
   (ii) A_i ∩ A_i = A_i
   (iii) A_i ⊆ A_i or A_i ⊇ A_i
   (iv) A_i ∩ A_i = ∅ and A_i ∉ A_i , and A_i ∉ A_i
   In case 1, it means agents can discard this node and consider the next one from the candidate list if it exists. Cases 2 and 3 are the most expected ones but they are indeed an ideal case. In this case, since it exactly matches, mapping function is 1:1 and no need to worry about semantics of them. However, we normally encounter case 4 where those two can only partially match. So we need to compute S_i = | A_i ∩ A_i | A_i ∪ A_i | (0 ≤ S_i ≤ 1) for each C_i against default value 1. Now for the specific nodes (i.e. C_i, C_i) we get S_i. From there going up to the root of the subgraph, updating default number 1 by applying function min() to each upper node in the target ontology.
5. After all candidates from the target ontology have finished step (3), we get a list in descending order by value S_i.
6. The biggest value S_i is the desirable one under certain circumstance (with a threshold). Since S_i → (A_i, A_i) → (C_i, C_i) , a mapping is found which indicates that a linkage exists from C_i → C_i. A path from the root down to target concept C_i is specified in the target ontology after mapping. At this stage, agents can transform...
source concept instance $t_i'$ into target concept instance $t_i''$ by referring to concept $c_i''$. In terms of benefits of ontology mapping to KB, it may add mapping rules into KB to further describe mapping related rules. It is in accordance with the ontology reuse even though it is the ultimate aim of ontology management.

5. Example

In our example, a particular agent will compute similarities between $C_S^{RMITCSIT.Student.Postgraduate.Postgraduate ResearchStudent.PhDstudent}$ and $C_T^{SwinIT.Student}$, $C_S^{RMITCSIT.Student.Postgraduate.PhDstudent}$ and $C_T^{SwinIT.Staff.ResearchStudent.PhDstudent}$ respectively. Suppose they have some lexical meanings, predefined rules are shown in the following none exhaustively.

```
;;rule
(defrule phd-holder
  phd-student
  => (is-phd-holder)

;;function signature
(defun have-fixed-room-number () (...) )
(defun have-fixed-phone-number () (...) )
(defun have-fixed-research-field () (...) )

;; start rule
(defrule start ""
  => (assert (is-phd-holder))
```

Following the discussion in Section 4 especially in step (3), we have $S_{DA} = |A_D \cap A_A| / |A_D \cup A_A| = 0.375$ for ($C_D \rightarrow C_A$) and $S_{DA} = |A_D \cap A_A| / |A_D \cup A_A| = 1$ for ($C_D \rightarrow C_A$) (see Figure 3). According to the algorithm, the agent will choose a mapping between source concept “PhDStudent” under RMITCSIT and a target concept under SwinIT (see Figure 3).

6. Related work

Of the related work in this area, some progresses have been made [4]. For instance, PROMPT [6] is a semi-automatic approach to ontology merging and alignment where users’ intervention is required. GLUE [1] applies machine-learning techniques to find ontology mapping by defining probability to several practical similarity measures. Our method is different from approaches of [1, 6] where we concentrate on ontology semantics conveyed by the taxonomy structure and inherent logic relations based on presumed common knowledge rather than providing tools. By doing so, agents are allowed to compute their individual similarities regarding certain criteria according to the domain theory. Initial rules are required if better choices are requested. It is a promising approach associated with ontology-related issues.

7. Conclusions and future work

In this paper, we introduce a multi-agent system architecture and an algorithm for multiple agents to manage and deploy ontologies in a dynamic environment. The architecture is derived from the proposed three level conceptual model. We demonstrate how this approach will enable ontology evolution and integration by using the JADE platform in our preliminary work. Our contributions lie in that the approach presented in this paper can be applicable to a broad range of common ontology-related problems, such as ontology integration and data translation among the ontologies - presumably conceptual models are constructed on top of common knowledge. Ontology mapping is in accordance with knowledge reusing and sharing conventions.

Although the approach in this paper is a promising way for ontology management or even further to semantic Web, many issues need to be further addressed. For example, how can ontology agents automatically or semi-automatically find similarities and differences between large scale ontologies effectively and efficiently? What is the tradeoff of ontology mapping? Moreover, one of the major challenges in this field is about representing uncertainty and imprecision in mapping.

Acknowledgement. Work reported in this paper is partly supported by Swinburne Vice Chancellor’s Strategic Research Initiative Grant 2002-2004 for project “Internet-based e-business ventures”.

References: