Dynamic Sub-Ontology Evolution for Collaborative Problem Solving

Yuxin Mao[1,2], William K. Cheung[1], Zhaohui Wu[2], Jiming Liu[1]

[1]Computer Science Department
Hong Kong Baptist University
Kowloon Tong, Hong Kong
[2]College of Computer Science
Zhejiang University
Hangzhou 310027, China
{maoyx@zju.edu.cn, william@comp.hkbu.edu.hk, wzh@zju.edu.cn, jiming@comp.hkbu.edu.hk}

Abstract

Emerging Web technologies have enabled the use of distributed on-line resources to support on-demand cooperative problem solving. Domain ontology is one of the resources typically required by many existing problem-solving systems. In this paper, we argue that to meet the on-demand requirement, on-line problem-solvers should go beyond the use of static domain ontology and be able to self-evolve and specialize in the knowledge they possess (called sub-ontology). Building on top of the Semantic Web technology, we propose an agent-oriented architecture and a sub-ontology evolution mechanism for dynamic self-organization of domain sub-ontologies to result in a truly intelligent and on-demand problem-solving platform in a distributed environment like the Web/Grid.

Introduction

Solving complex domain-specific problems usually requires the interoperation of a domain-independent reasoning unit, a set of contextually related application components and a corresponding set of (normally large-scale) domain knowledge. Emerging Web technologies have enabled the conventionally centralized problem-solving approaches (e.g., problem-reduction, planning, etc.) to take advantage of distributed dynamic resources to support cooperative problem solving [Benjamins, 1999; Wooldridge & Jennings, 1999; Martinez & Lesperance, 2004]. By modeling the application components (implemented as information or knowledge services) as operators and the solution as the final goal, the problem-solving process, to a certain extent, can be formulated as an AI planning problem [Blythe et al., 2003].

Many existing planning-based problem-solving systems [Carver & Lesser, 1993; Lopez & Plaza, 1993; Durfee, 2001] assume that either a complete domain knowledge is available for planning the problem-solving process or it can be acquired via user interaction during the planning. However, this assumption fails for dynamic environments like the Web/Grid [Foster and Kesselman 1999, Foster et al, 2001]. First, it could be too costly for acquiring and maintaining a large variety of domain knowledge for solving problems of various types. Second, some knowledge could only be or much easier to be *discovered* based on past planning (problem-solving) experience, instead of designed by domain experts. Third, many problems are cross-domain and maintaining the corresponding customized knowledge is nont-rivial.

As echoed in [Jennings, 1992], sophisticated problem solving requires *knowledge about the problem domain and knowledge about problem solving per se*. Recent advent of Web ontology, as the foundation of the Semantic Web [Berners-Lee, 2001], has facilitated the incorporation of different types of on-line ontology. There have been a few of ontologies for problem solving [Ikeda, 1997; Fensel & Benjamins, 1999; Crubezy & Musen, 2003; OWL-S, url]. As OWL [OWL, url] has become the *de facto* standard of the Web ontology, we will mainly refer to domain knowledge in terms of OWL ontologies.

In this paper, we will focus on the role of *knowledge about the problem domain* in the process of problem solving and how the knowledge can evolve to support efficient problem solving. We propose, on top of the Semantic Web technology, an agent-oriented architecture, which adopts a local sub-ontology evolution mechanism for dynamic self-organization of domain knowledge to support intelligent and efficient planning for problem solving in a distributed environment like the Web/Grid.

The remaining of the paper is organized as follows. In section two, we first introduce the concept of sub-ontology evolution briefly. Then we give the formal specification of ontology representation in section three. The essential operations for sub-ontology manipulation are the topic of section four. We present an agent-oriented architecture with a local sub-ontology evolution mechanism to support intelligent and efficient problem solving in sections five.

We demonstrate the proposed sub-ontology evolution mechanism can be applied to problem solving with a Traditional Chinese Medicine (TCM) use case in section six. Section seven gives an overview of related works and section eight concludes the paper with an outlook to future research directions.

Sub-Ontology Evolution

By formulating problem solving as planning, the process can intuitively be understood as consecutive decomposition of a goal into sub-goals, forming a task graph eventually. In the Web community, this process is typically called service composition. During each of the planning steps, domain knowledge is often needed for, say, service profile and process matching so as to identify suitable service agents to fulfill the sub-goals. Thus, problem-solving agents will have their problem-solving power restricted if the domain knowledge they possess is insufficient. Including all the ontologies in their complete form could of course solve the problem but imply unnecessarily huge storage and computational requirement.

Our conjecture is that an agent specialized in a certain underlying sub-problem needs only particular aspects of the whole ontology. This calls for the agent's capability to extract from a large ontology (e.g., Gene Ontology [1]) specific portions and keep *evolving* them to form a knowledge base as *sub-ontologies* (SubO), emerged to be specialized in its own areas of concern. We believe that this capability is especially important for cross-domain problems where multiple ontologies corresponding to knowledge of several domains are typically required. Here we define *sub-ontology evolution* as follow:

> The sub-ontology evolution of a problem-solving agent is defined as the process which involves autonomous extraction, merging, mapping and integration of some sub-structures of the agent's previously encountered domain ontologies, termed as sub-ontology, into its own localized repository of domain knowledge to support intelligent and efficient problem solving.

To contrast with related areas like ontology engineering, our concern of ontology evolution is inclined to solve domain-specific problems via highly dynamic update of the sub-ontologies possessed by different agents, where the sub-ontologies are emerged as by-products of the process of problem solving. While our proposed methodology can still benefit from those algorithms for ontology engineering (extracting, learning, merging, etc.), we propose to extend from the sub-ontology extraction to dynamic sub-ontology evolution in a distributed environment using past problem-solving history.

[1] http://www.geneontology.org/

Formal Representation

An ontology always contains relatively complete domain knowledge. As before mentioned, problem solving is always local to a subset of known information like a large-scale ontology. According to [Ghidini & Giunchiglia, 2001], the part of what is potentially available being used while reasoning is what we call *context* (of reasoning). In problem solving, particular aspects of the complete knowledge can be reused under certain contexts and those context-specific sub-structures of the ontology are treated as sub-ontologies. In this section, we illustrated how to represent ontology and sub-ontology formally based on the semantic structure of ontology.

Semantic Graph

For a large domain ontology, the relations among semantic entities are anfractuous, which form the complex structure of domain ontology. Graph theory [Harary, 1972] as a branch of mathematics has applications in many areas and many real-world problems can be modeled using graphs. If we view a large domain ontology as a complex directed and weighted graph [Mitra, 2000], we can adopt the graph theory to represent its semantic structure from the graph perspective.

Definition 1 (Semantic Graph). A semantic graph is the graph representation of a domain ontology, as a triple $<O, N, A>$, where O is the domain ontology, N is a set of nodes for concepts in O, and A is a set of weighted arcs for relations.

Each node v is associated with a concept c and in semantic graph a concept is always linked to several other concepts by arcs specifying the relations between two concepts. Each arc e is associated with one or more relations. The domain of a relation is denoted by one end of e and the range is denoted by the other end (the domain concept and the range concept are related by the relation). If there are more than one relation between two concepts, in order to express how many relations an arc represents we associate each arc with a relational weight $w(e)$.

Although an OWL ontology itself is a graph, the major difference between semantic graph and the graph model for OWL ontologies is that Semantic Graph adopts the attributes and invariants of graph theory which will be computable for further manipulations on ontology.

Definition 2 (Semantic Distance). A path in a semantic graph G is a non-empty subgraph $P = <N, A>$ of the form $N = \{v_0, v_1, \ldots, v_k\}$, $A = \{ v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\}$, where the v_i are all distinct (according to the definition of Path in graph theory). If P is a shortest path from v_0 to v_k in G, the semantic distance between v_0 and v_k can be computed by the following equation:

$$d_G(v_0, v_k) = 1/w(v_0v_1)+1/w(v_1v_2)+\ldots+1/w(v_{k-1}v_k).$$

If there are more relations between two concepts, it implies a closer relationship of the two concepts, so an arc with a higher relational weight means shorter distance in a path. A long semantic distance illustrates a distant relationship between two concepts in ontology.

Sub-Ontology

We can give a formal definition for sub-ontologies based on the semantic graph representation of ontology.

Definition 3 (Sub-Ontology). Formally, a sub-ontology (SubO for short) is a triple $<C, S, O>$, where C is a set of focusing concepts denoting the *context* of problem-solving, S is a self-contained and closed set of facts and axioms *relevant* to C, and O is a set of *links* to the SubO's original domain ontologies.

Focusing concepts are not always the top-concepts in the concept hierarchy tree of a SubO, but the concepts related with a problem-solving context at most. A set of focusing concepts are used to capture the features of the problem-solving context. Focusing concepts can vary with contexts and it makes sense that a SubO can be reused in different contexts. The self-contained and closed characteristic of S means that all information reference is involved in S. The contents of S can be represented as one or more subgraphs in the semantic graph of the original ontology.

Sub-Ontology Manipulation

Based on the formal specification in previous section, we propose a set of operations for manipulating SubOs from domain ontologies.

Traverse. The traversing operation can be defined as a tuple $T = <c, R, n, G>$, where c is the starting concept for traversal, n is the depth limit, R is a set of constraint relations and G is a *link* to the semantic graph of the target domain ontology. The operation will return a set of facts and axioms S from the ontology.

We combine all the semantic entities that can support the explanation of a concept in *direct knowledge*:
Definition 4 (Direct Knowledge). Given an ontology concept c, its direct knowledge $D(c)$, is a set of semantic entities including related concepts with corresponding relations, extending individuals and other attributes of c.

The process of traversal can be treated as traversing the semantic graph of an ontology:

1. First, starting at the concept c, trace in each direction along its adjacent edges that contain relations appeared in R.

2. The traversal will fall into several branches. For each concept on a traversing path, its direct knowledge is allocated into a set, if not in.

3. Each traversing branch or path terminates when a depth n of semantic distance has been met. Assume c_l is the last concept in a traversing path and the concepts in $D(c_l)$ are called *edge concepts* whose definition may not be allocated in completely.

For example, if $R = \{rdfs:subClassOf\}$ and $n = 2$, the traversal will get a class hierarchical tree with depth 4 (two levels upward and two levels downward). R can be set to *All* that will include all relations in the ontology.

Extract. The extracting operation can be defined as a tuple $E = <C, R, n, G>$, where C is a set of focusing concepts, n is the depth of the extraction, R is a set of constraint relations and G is a *link* to the semantic graph of the target domain ontology. The operation will return a SubO, B.

The process of extraction can be treated as a group of traversing operations. For each of the focusing concepts, perform a traversing operation and combine all resulting sets together. Except edge concepts, concepts whose direct knowledge is not completely included in the set will be eliminated from the set. The final set is just the self-contained set of a SubO.

To extract SubOs from an ontology, we must first identify focusing concepts, which denotes the context for problem solving to the complete ontology. However, transforming a natural user description to the explicit semantic description of a context falls into the fields of human computer interaction and natural language processing, and, due to the focus of this paper, will not be further discussed, so we assume the focusing concepts have already been identified before extraction.

Augment. The augmenting operation can be defined as a tuple $A = <C', S', B>$, where C' is a set of new focusing concepts to be added to C, S' is a set of facts and axioms from the same domain ontology, and B is the original SubO. The operation will return an updated SubO, B_a.

The additional set of focusing concepts and set of contents from the same domain ontology are simply integrated into the original SubO. However, the self-contained set of the augmented SubO has to be verified to confirm that it's still closed and self-contained and any contents lead to inconsistency will be discarded.

Prune. The pruning operation can be defined as a tuple $P = <C', B>$, where C' is a list of focusing concepts to be eliminated from C, and B is the original SubO. The operation will return an updated SubO, B_p.

After a list of focusing concepts has been removed from the original SubO, the self-contained set has to be trimmed: any contents that can be allocated by traversing from the concepts in the list should be removed from the set.

Refocus. The refocusing operation can be defined as a tuple $F = <C', B, G>$, where C' is a new set of focusing concepts to the SubO, B is the original SubO, and G is a *link* to the semantic graph of the original domain ontology. The operation will return an updated SubO, B_f.

The process of refocusing can be formalized as the following equations:

$$B_p = P <B.C - C', B>$$
$$C_a = C' - (B.C \text{ and } C')$$
$$B_e = E < C_a, B.R, n, G>$$
$$B_f = A < C_a, B_e.S, B_p>$$

Refocusing means the problem-solving context for a SubO has changed, which is perhaps the most complex manipulation in the model. If the focusing concepts of a SubO changed, the contents of the corresponding self-contained set need to change (augment, prune or reorganize) according to the new focusing concepts.

Compare. The comparing operation can be defined as a tuple $C = <B_1, B_2, G>$, where B_1 and B_2 are two SubOs to be compared, and G is a *link* to the semantic graph of the original domain ontology. The operation will return a degree of similarity between the two SubOs.

The comparison can also be performed based on the edit distance or the Levenshtein distance [Levenshtein, 1966]. The similarity between strings is often described as the edit distance, the number of changes (deletions, insertions, or substitutions) required to turn one string into another. However, in this operation, we don't want to compare strings but concepts. There are two strategies for comparing two SubOs. One is based on the edit distance of two list of focusing concepts, which is the number of changes required to turn one list of concepts into another list. The other is based on the edit distance of two self-contained sets. Therefore, we calculate the number of deletions, insertions, or substitutions operations of semantic entities needed to turn one set into another. The former is simpler, but the latter is more exact.

Merge. The merging operation can be defined as a tuple $M = <B_1, B_2, G>$, where B_1 and B_2 are two SubOs to be merged, and G is a *link* to the semantic graph of the original domain ontology. The operation will return a new merged SubO, B_m.

The process of ontology merging is building an ontology by merging several ontologies into a single one that unifies all of them [Pinto, 1999]. Although researchers have worked on automatic or tool-supported merging of ontologies [Stumme & Maedche 2001; Noy & Musen, 2000], ontology merging is still in its early stages. The most difference between merging operation for SubOs and existing ontology merging methods is that the parts of SubOs to be merged come from the same ontology, so conflicts and differences are much fewer.

We can simply combine the self-contained sets of each SubO together to get S of B_m. It's trivial to prove the conjunction of self-contained sets is still self-contained, but probably with redundancy. There must be some mechanisms to eliminate the redundancy. We adopt part of the method proposed in [Noy & Musen, 2003] to merge SubOs descending from the same domain ontology.

merge concepts. If two concepts C_1 and C_2 have the same identifier, create a new concept C_m in S of B_m. For each entity in $D(C_1)$ or $D(C_2)$, create an image in $D(C_m)$. If either C_1 or C_2 was already in S prior to the operation, all references to it in B_m become references to C_m and the original concept is just deleted.

merge relations. As relations in Web ontology languages like OWL are first class object [Berners-Lee, 2001], merging relations is similar with merging concepts. If two relations R_1 and R_2 have the same identifier, create a new relation R_m in S of B_m. For each concept C in the domain and range of R_1 or R_2, if there is an image of C in S, add it to the domain or range of R_m, respectively. If either R_1 or R_2 was in S prior to the operation, all references to it become references to R_m and the original relation is deleted. Note if there was a concept in S that had both R_1 and R_2 attached to it, after the operation it will have only R_m attached to it.

As merge other entities in the direct knowledge of a concept is almost the same as merging relations, it's no need to describe them in detail.

Sub-Ontology Evolution for Problem Solving

We envision a distributed problem-solving environment with many application components as service agents. The service agents include: (1) agents that can provide some specific information or functionalities as described in their service profiles (classified into information-gathering service and world-altering service in Web Service); (2) agents that provide access to large-scale domain ontologies; and (3) problem-solving service agents (or simply problem-solving agents) that can *compose* and *coordinate* a set of service agents for solving problems. Among them, the latter two are our focus as they characterize how well the service composition can be supported given a distributed environment.

An Agent-Oriented Architecture

As mentioned in the previous subsection, two key components in our proposed agent-oriented architecture are:

Domain Ontology Service Agents (DO-Service). Each domain ontology service agent preserves a large-scale domain-specific ontology (DO) and provides a set of basic inquiry operations on the ontology. Note that we assume the DO is invariant, instead of being dynamically updated as Local-KB. However, DO-Service can provide various semantic views [Wu, 2004] to knowledge consumers.

Problem-Solving Service Agents (PS-Service). Each problem-solving service agent in the proposed problem-solving environment maintains a locally centralized knowledge base (Local-KB) with sub-ontologies of different domains. There is also a domain independent planner which should be able to work with DO-Services to compose services in a goal-directed manner, where a problem-solving ontology (PO) is installed in the PS-Service which holds the knowledge about problem-solving per se, which can be represented by some extended version of OWL-S like Problem-Solver Mark-up Language (PSML) [Liu, 2003].

Problem Solving with Sub-ontology Evolution

The details about how to establish a plan are not the focus of this paper, which we believe could be achieved by existing planning methods. Our main objective is to illustrate how different aspects or views of large-scale domain ontologies, with the help of past collaborative problem-solving history, can naturally diffuse to a set of distributed PS-Services; each becomes specialized in solving some sub-problems via self-evolution.

The following is the detailed process (see figure 1) of the proposed sub-ontology evolution approach for problem solving:

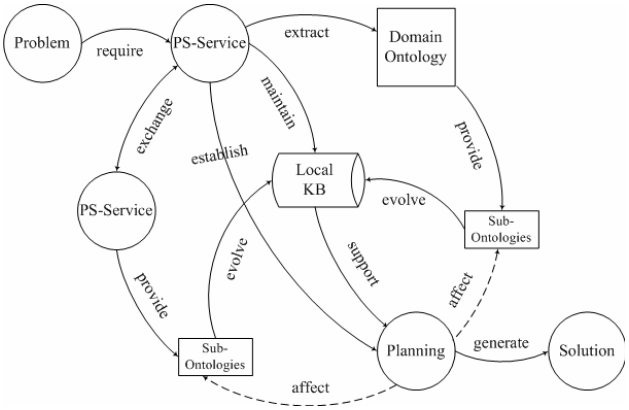

Figure 1. A possible process of sub-ontology evolution in problem solving.

1. A PS-Service first selects and extracts domain-specific SubOs in Local-KB according to the focusing concepts of the current context. The SubOs are used to support subsequent semantic service matching based on the "Service Profile" and the "Service Process" descriptions. Transforming domain knowledge into knowledge exactly needed by planning may be involved.

2. If no appropriate SubOs in the Local-KB can be **reused**, the agent will turn to its adjacent PS-Service agents (based on some P2P strategy) to look for relevant domain knowledge that could be distributed in the Web/Grid.

3. If the aforementioned P2P knowledge search succeeds, it means that there exists at least one adjacent PS-Service that is specialized in solving the current problem and the planning steps can just proceed based on matching results.

4. If the P2P search fails, it hints lacking relevant domain knowledge in the agent's proximity. The PS-Service will try to identify a relevant DO-Service agent directly to support the semantic service matching. SubOs potentially needed for the matching will be **extracted** accordingly from the DO of the DO-Service.

5. The composed service plan/flow obtained with the help of the SubOs will then be executed. If the execution performance is positive, the P-Service will evolve its Local-KB by integrating the new SubOs (extracted from the distributed sources) into its Local-KB. During the integration, new-coming SubOs are **compared** with the existing ones to see if there exist closely relevant SubOs for **merging**. The part of a SubO that is not used for supporting the planning will be **pruned** or **augmented**. The SubOs not used at all for the planning will simply be discarded.

As the SubOs of different PS-Services continue to evolve, a community of specialized PS-Services will emerge. Faced with similar planning contexts next time, those PS-Services can solve problems based on reusing the existing SubOs in their Local-KB immediately. Note that consulting first the adjacent PS-Services is to maximize the reuse of existing SubOs and in the long run to result in a system efficient enough on average for serving on-demand problem-solving requests.

A TCM Use Case: Herb Recommendation

We can simulate a distributed agent-oriented environment for TCM problem solving: there are many service agents that can provide specific information about TCM and services are annotated with semantic descriptions in service profile and service process; there is a large-scale TCM domain ontology called TCMLS (Traditional Chinese Medicine Language System) [Zhou, 2004], which is the world's largest TCM ontology including 20,000

concepts and 100,000 individuals in current knowledge base. The TCMLS consists of several distributed TCM **sub-domain** ontologies (e.g., Formula, TCM Basic Theory, TCM Literature), with each wrapped as a DO-Service; there are several planning service agents that can compose and coordinate a set of service agents for solving TCM domain problems.

A TCM **formula** of TCM is composed of one or more **Chinese medicinal herbs**. It is a carefully assembled combination of **herbal components**, each having its **functions** and mixed together in proper **amount**s in accordance with specific principles for combining herbs. To describe how the proposed methodology works in practice, let's envision the following **scenario**: a TCM **chemist** wants to look for some **Chinese medicinal plants**, which can be potentially used to **compose** a **new formula** that can treat a set of **diseases**.

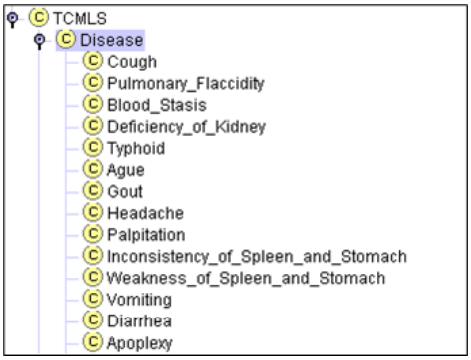

Figure 2. A sub-ontology focusing on disease.

The chemist submits its problem to the environment and the task is automatically assigned to a relevant PS-Service **A**. **A** first selects a rough collection of service agents that provide specific information about Chinese medicinal herb, formula and disease. According to a disease SubO **<{Disease}, S_d, O_d>** (see figure 2) in its Local-KB, **A** picks out a set of service agents that provide explicit information about disease individuals (see figure 3). O_d denotes the link to the Disease sub-domain ontology of the TCMLS. Note that we have translated Chinese characters into English for explanation, and only partial view of ontology is shown in figures.

```
<process:AtomicProcess rdf:ID="DiseaseProcess">
        <process:hasInput rdf:resource="#Description"/>
        <process:hasOutput rdf:resource="#Disease"/>
</process:AtomicProcess>
<process:Input rdf:ID="Description">
         <process:parameterType rdf:datatype="&xsd;anyURI">
              &xsd;string
        </process:parameterType>
</process:Input>
<process:Output rdf:ID="Disease">
        <process:parameterType rdf:resource="&tcmls;Disease"/>
</process:Output>
```
Figure 3. A segment of process model description of disease service.

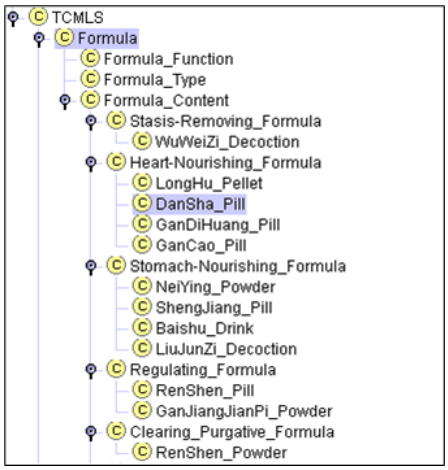

Figure 4. A sub-ontology focusing on formula**.**

Since **A** lacks domain knowledge about Chinese medicinal herb and formula, it then turns to an adjacent PS-Service **B** for help. **B** happens to be specialized in formula recommendation and exchanges a corresponding SubO **<{Formula-Content}, S_f, O_f>** (see Figure 4) with **A** according to its requirements. As the concepts **Formula** and **Disease** are related by the relation **treat** according to the SubO in figure 4, **A** can match output of the service agents in figure 3 with the input of the service agents in figure 5 and compose them together according to the SubO.

```
<process:AtomicProcess rdf:ID="FormulaComponentsProcess">
        <process:hasInput rdf:resource="#Formula"/>
        <process:hasOutput rdf:resource="#Chinese_Medicinal_Herb"/>
</process:AtomicProcess>
<process:Input rdf:ID="Formula">
        <process:parameterType rdf:resource="&tcmls;Formula"/>
        <rdfs:label>Input Formula</rdfs:label>
</process:Input>
<process:Output rdf:ID="Chinese_Medicinal_Herb">
         <process:parameterType
         rdf:resource="&tcmls;Chinese_Medicinal_Herb"/>
        <rdfs:label>Output Chinese Medicinal Herb</rdfs:label>
</process:Output>
```
Figure 5. A segment of process model description of formula components service.

Composing the service agents in figure 3 and figure 5, we can get a list of Chinese medicinal herbs, which may be candidates for the new formula. The last step is to picks out the herbs that are medicinal plants; however, both **A** and **B** as well as other adjacent PS-Services have no knowledge about Chinese medicinal herbs. **A** then finds that there is a DO-Service **D** with the Chinese Medicinal Herb sub-domain ontology and extracts a SubO **<{Chinese-Medicinal-Herb}, S_h, O_h>** (see Figure 6) from **D**. According to the semantic relations in the SubO, **A** can match service agents in figure 5 with the ones in figure 7.

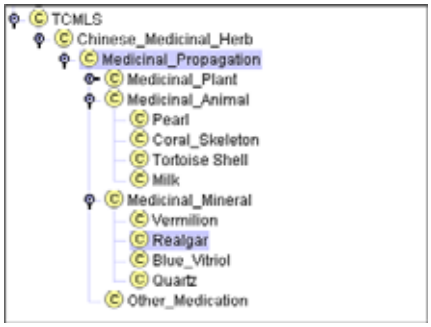

Figure 6. A sub-ontology focusing on medicinal propagation.

```
<process:AtomicProcess rdf:ID="HerbTypeProcess">
        <process:hasInput rdf:resource="#Chinese_Medicinal_Herb"/>
        <process:hasOutput rdf:resource="#HerbType"/>
</process:AtomicProcess>
<process:Input rdf:ID="Medicinal_Propagation">
         <process:parameterType
         rdf:resource="&tcmls; Medicinal_ Propagation "/>
        <rdfs:label>Input Medicinal  Propagation </rdfs:label>
       </process:Input>
<process:Output rdf:ID="HerbType">
         <process:parameterType rdf:datatype="&xsd;anyURI">
              &xsd;string
        </process:parameterType>
        <rdfs:label>Output Herb Type</rdfs:label>
</process:Output>
```

Figure 7. A segment of process model description of herb type service.

After completing the whole plan with a set of Chinese medicinal herbs recommendation, the Local-KB of **A** will evolve with both the formula SubO and the Chinese medicinal herb SubO that have supported the problem-solving process. New-coming SubOs can be merged with the existing one as a new SubO **<{Disease, Formula-Content, Medicinal-Propagation}, S, {O_d, O_f, O_h}>** (see figure 8). The part of a SubO that is not used for supporting the planning will be pruned. For example, during the planning process, **A** found that the herbs were of either **Medicinal-Plant** or **Medicinal-Mineral,** so the portion of knowledge about **Medicinal-Animal** can be discarded. **A** now becomes a bit more specialized in herb recommendation.

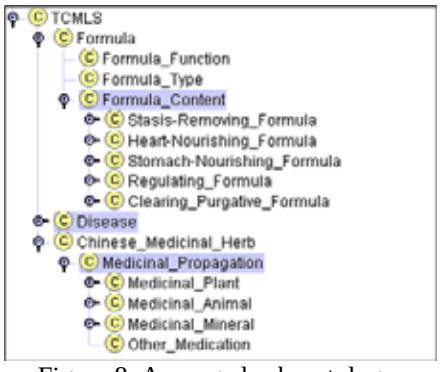

Figure 8. A merged sub-ontology.

Related Work

Up to now, there exist little similar on-going research effort reported in the literature, compared with our work; however, there are a few of related works in several fields. Semantic-based or planning-based service composition [Paolucci, 2001; Li & Horrocks, 2003] is proposed to meet on-demand requirements. There is also some research effort on extracting particular aspects or views of a domain ontology [Noy & Musen 2004; Bhatt & Taniar, 2004].

Conclusion

In this paper, we proposed an agent-oriented architecture and a sub-ontology evolution mechanism for collaborative problem solving in a distributed environment like the Web/Grid and illustrated how it could work in practice via a TCM use case. A formal specification of sub-ontology as well as a set of manipulations is given in detail. We believe this self-evolutionary paradigm brings us a step closer to on-demand problem solving.

Future research issues include (1) how to map portions of domain ontologies to services efficiently and accurately, (2) how to integrate SubOs into the Local-KB in some optimal sense and hierarchically organize SubOs in the Local-KB to support efficient planning, and (3) how to maintain the consistency between local SubOs and remote DO.

Acknowledgement

This work was partially supported by a grant from RGC Central Allocation Group Research Grant (HKBU 2/03/C), a grant from the Major State Basic Research Development Program of China (973 Program) under grant number 2003CB316906, a grant from the National High Technology Research and Development Program of China (863 Program): TCM Virtual Research Institute (sub-program of Scientific Data Grid), and also a grant from the Science and Technique Foundation Programs Foundation of Ministry of Education of China (No. NCET-04-0545).

References

Benjamins, V.R. et al. 1999. *Towards Brokering Problem-Solving Knowledge on the Internet.* Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW 1999) 33–48.

Wooldridge, M., and Jennings, N.R. 1999. *Cooperative Problem Solving Process.* Journal of Logic & Computation 9(4).

Martinez, E., and Lesperance, Y. 2004. *Web Service Composition as a Planning Task: Experiments using Knowledge-Based Planning.* Proceedings of the ICAPS-2004 Workshop on Planning and Scheduling for Web and Grid Services.

Blythe, J. et al. 2003. *The role of planning in Grid environment.* Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003).

Carver, N., and Lesser, V. 1993. *A Planner for the Control of Problem-Solving Systems.* IEEE Transactions on Systems, Man, and Cybernetics, special issue on Planning, Scheduling and Control 23(6).

Lopez, B., and Plaza, B. 1993. *Case-based Planning for Medical Diagnosis.* Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems(ISMIS 1993) 96–105.

Durfee, E.H. 2001. *Distributed Problem Solving and Planning.* Mutli-agents systems and applications, Springer-Verlag New York, Inc., New York, NY.

Foster, I., and Kesselman, C. 1999. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann.

Foster, I., Kesselman, C., Tuecke, S. 2001. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations.* International Journal of High Performance Computing Applications Vol. 15 pp.200–222.

Jennings, N.R. 1992. *A Knowledge Level Approach to Collaborative Problem Solving.* Proceedings of 10th European Conference on Artificial Intelligence (ECAI 1992).

Berners-Lee, T., Hendler, J., and Lassila, O. 2001. *The Semantic Web.* Scientific American 284(5) 34–43.

Ikeda, M., Seta, K., Mizoguchi, R. 1997. *Task Ontology Makes It Easier To Use Authoring Tools.* Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1997) 342–351.

Fensel, D., and Benjamins, V.R. 1999. *UPML: A Framework for Knowledge System Reuse.* Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1999) 16–23.

Crubezy, M., and Musen, M.A. 2003. *Ontologies in Support of Problem Solving.* Handbook on Ontologies 321–341.

OWL-S, url. Available: http://www.daml.org/services/owl-s/1.1.

OWL, url. Available: http://www.w3.org/2001/sw/WebOnt/.

Ghidini, C. and Giunchiglia, F. 2001. *Local Models Semantics, or Contextual Reasoning = Locality + Compatibility*. Artificial Intelligence, Vol. 127 pp.221–259.

Harary, F. 1972. *Graph Theory*. Reading, Massachusetts: Addison-Wesley.

Mitra, P., Wiederhold, G., and Kersten, M.L. 2000. A *Graph-Oriented Model for Articulation of Ontology Interdependencies*. Proceedings of the 7th International Conference on Extending Database Technology (EDBT 2000).

Levenshtein, I.V. 1966. *Binary Codes Capable of Correcting Deletions, Insertions, and Reversals*. Cybernetics and Control Theory 10 (8) 707-710.

Pinto, H.S., Gomez-Perez, A., Martins, J.P. 1999. *Some Issues on Ontology Integration*. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1999).

Stumme, G., and Maedche A. 2001. *FCA-MERGE: Bottom-Up Merging of Ontologies*. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2001).

Noy, N.F., and Musen, M.A. 2000. *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-2000).

Noy, N.F., and Musen, M.A. 2003. *The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping*. International Journal of Human-Computer Studies 59 (6).

Wu, Z., Mao, Y., and Chen, H. 2004. *Semantic View for Grid Services.* Proceedings of 2004 IEEE International Conference on Services Computing (SCC 2004) 329–335.

Liu, J. 2003. *Web Intelligence (WI): What makes Wisdom Web?* International Joint Conference on Artificial Intelligence (IJCAI 2003), Invited Talk.

Zhou, X., Wu, Z., Yin, A. et al. 2004. Ontology Development for Unified Traditional Chinese Medical Language System. Journal of Artificial Intelligence in Medicine, 32(1) 15–27.

Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K. 2002. *Semantic Matching of Web Services Capabilities.* Proceedings of 1st International Semantic Web Conference (ISWC 2002).

Li, L., and Horrocks, I. 2003. *A Software Framework for Matchmaking Based on Semantic Web Technology.* Proceedings of the 12th International World Wide Web Conference (WWW 2003).

Noy, N.F., and Musen M.A. 2004. *Specifying Ontology Views by Traversal.* Proceedings of the 3rd International Semantic Web Conference (ISWC 2004).

Bhatt, M., and Taniar, D. 2004. *A Distributed Approach to Sub-Ontology Extraction.* Proceedings of 18th International Conference on Advanced Information Networking and Applications (AINA 2004).