Petri Net based Approach for Merging Ontologies in Web Service Composition Scenario

Debajyoti Mukhopadhyay, Archana Chougule*

Department of Information Technology, MIT, Pune -38, India

Abstract

Ontologies are being used for construction of semantically rich service descriptions. In case of Web service composition, techniques for planning, composing, editing, reasoning and analyzing of these descriptions are required. It’s necessary to resolve semantic interoperability between services to achieve Web services coordination and collaboration. Matchmaking and merging of service descriptions is one of the key issues in semantic Web because it is the basis of doing Web service discovery and composition. In this paper we explore a method to merge service descriptions which are prepared as ontologies. We analyze ontologies describing processes related with the Web services using Petri Nets. We take as our starting point, the OWL-S Process Models for describing the Web services capabilities. We define workflow and sequence of processes descriptions for merging ontologies related to Web services participating in the composition. We represent each process description in OWL-S as Petri Net model and show how integrated Petri Net helps to complete ontology merging process. This enables performance improvement in Web service composition process and also helps in verifying the Web service composition.

Keywords: Semantic Web, ontologies, Petri Nets, Web service composition, ontology merging, OWL-S

1. Introduction

Web service composition plays an important role to support in enterprise applications integration and business to business communication. It provides a mechanism for integrating heterogeneous applications required for business by connecting related Web services together. During the procedure of Web service composition, most of the time goes in managing exceptions and transaction integrity. Different standards like BPEL4WS, WSMO, BPML and WSCI are available to achieve Web service composition. In composition procedure, description of Web services is required to be provided. In Semantic Web community, reasoning about Web resources is provided by explicitly declaring their preconditions and effects with terms precisely defined in ontologies. One of the languages used for Web service description is OWL-S.

Workflow of processes in Web services is described in OWL-S Process Model. It is one of the leading standards for the description of Web services in the Semantic Web. The OWL-S Process Model describes the interaction protocol between a Web service and its clients which may use the Web service for composition. For each process, there are three components: inputs, preconditions and results. Outputs and effects produced by the process under a given condition are specified as results. The processes in OWL-S describe information transformation produced by the Web service. Preconditions and effects in OWL-S describe the state transition of knowledge which is produced by the execution of a Web service.

Many times we need to merge these OWL-S Process Models while carrying out Web service composition. In this paper we focus on defining the sequence of processes descriptions to be merged. We analyze and verify the inputs, preconditions and outputs of the processes in OWL-S with the help of Petri Nets. We design an integrated Petri Net which represents all processes involved in Web service composition. We define the ontology merging process by considering atomic process sequences in the integrated Petri Net.

Rest of the paper is structured as follows: we first describe the work done by other researchers regarding ontology based Web service composition in Section 2. Web services composition framework for providing

* Corresponding author. E-mail: chouguleab@gmail.com
precomposed Web services is described in Section 3. Our approach for merging OWL-S Process Models is described in Section 4 with example. Section 5 gives implementation details. Impact analysis of our approach is mentioned in Section 6. We conclude the paper in Section 7.

2. Literature Review

Various approaches are suggested by researchers to carry out Web service composition using domain ontologies and service ontologies. G. Li, S. Deng, H. Xia and Chun Lin [1] implement Semi-automatic Web service composition using work flow technology. They make use of Business Process Execution Language and process ontology. They provide an algorithm to extract edge-edge and node-node relations from process ontology. They have built a process ontology tree using work flow technology. In process ontology tree, each service may correspond to a node of process ontology tree. A process ontology tree is used to extract all rules and relations applied to complete automated composition. After that, the target nodes are searched in process ontology tree and validation of interrelations is carried out. YU Qing-mei, XIAO Peng-yan and JIN Ting [2] suggest a heuristic approach for Web service composition, where it finds similarity based on domain ontology. A Web service composition graph is constructed and then a serial of composition path is generated using directed acyclic graph. A heuristic function is introduced which reduces the searching area. An algorithm is explained in the paper which considers services semantic similarity and adjusts composition plan dynamically. Jiangang Ma, Yanchun Zhang and Minglu Li [3] explain ontology based approach for Web service composition. The approach is goal driven. The architecture decomposes the user’s goal to sub goals. Domain specific ontology is used to annotate the information in the goal and Web services. They have also used AI technology and theory of reasoning about action to complete Web service composition task. Yajuan Song and Lei Liu [4] present a two way Web service composition method. Domain ontology is processed for dynamic Web service composition. They annotate domain ontology with the input and output parameters of the Web service. Parameters are denoted according to the concepts in domain ontology they belong to. Algorithms are provided to find input candidates, output candidates and to carry out web service composition process. Extension for OWL-S ontology framework to achieve dynamic web service composition is provided by Jing Dong, Yongtao Sun and Sheng Yang [16]. They provide new constructs for specifying higher level of abstraction. The approach includes a service instance pool. With the help of this pool, candidate services are filtered and plugged in at runtime. The paper provides planner prototype which is based on Java Theorem Prover(JTP). Three new constructs are defined in OWL-S process ontology namely AbstractProcess, abstractComposedOf and ControlConstruct. Authors detail how goals of customer’s are converted into RDF entries. These entries are then fed into the JTP inference engine.

Many researchers have proposed approaches for merging ontologies. Some of the approaches are as follows: Rajesh Kumar Gupta and B. B. Chaudhary [5] use description logic and description graph to merge domain ontologies. They have defined mapping rules and provided algorithm for merging which uses knowledge schema in TBOX and named entities in ABOX. Hong Xia, Zengzhi Li, Hai Wang and Yu Gui [6] provide a concept lattice prototype union algorithm which utilizes attribute based incremental formation algorithm. They construct a sub concept lattice. Then in turn insert concept of sub lattice into another sub lattice and finally form required concept lattice. Julia Taylor, Daniel Poliakov and Lawrence Mazlack [7] provide a method for merging domain specific ontology using a search engine combined with lexical, semantic and rule-based methods. The concepts are merged only if the similarity values are above a user-defined threshold. Lexical matching is done by identifying components with similar names and semantic matching is carried out using search engine. Matching rules used the nodes of the ontology which are simple Hearst patterns and propositional formulas. A model based on concept lattice for merging heterogeneous ontologies automatically is provided by Shixiang Li and Hong Fan [8]. The semantic similarity is calculated by using instance, definition and structure similarity. They describe that relations between local ontologies are found and merged. Global ontology is prepared then. Alm delia and Adolfo Guzman [9] propose an ontology merging notation that provides substantial improvements to the ontology languages like RDF and OWL. They also suggest an algorithm for automated merging of ontologies which uses similarity search functions applied to concepts. An ontology merging method based on WordNet mapping is suggested by Hyunjang Kong, Myungwoon Hwang and Pankoo Kim [10]. In the merging process, first WordNet mapping is done which is a preprocessing step for a more efficient similarity measurement. Set of candidate concepts is selected before measuring similarity between concepts. Similarity is measured using Jaccard coefficient and MSP(most-specific-parent method). After measuring similarity, ontology hierarchy is reconstructed. Target driven merging of ontologies is proposed by Salvatore Rannich and Erhard Rahm [11]. The approach is based on an equivalence matching between a source and target taxonomy to merge them. The approach preserves the structure of the target taxonomy as much as possible. The approach utilizes additional relationships between source and target concepts to semantically improve the merge result. Bartley Richardson and Lawrence Mazlack [12] use thesaurus files to measure lexical similarity between ontologies. The approach determines the decision to merge by measuring number of similar items in each ontology and thesaurus file. Probability of merge is computed by Bayes theorem and cosine and KL divergence functions.
3. Composite Web services framework

Here we discuss different modules of the framework designed for semi automated Web service composition [14]. We have designed a framework for providing recomposed Web services access for registered users. We assume that descriptions for all Web services are provided in OWL-S format.

![Diagram of composite web services framework]

Figure 1 shows different modules in the framework. First module in the framework is User Subscription Module. Each new user is required to register with the system, which helps to provide customer specific composite Web services facility. A unique identity number is assigned to each user. User can access any of the services listed under CompoServRegistry after completing registration process. Second module of framework is CompoServRegistry. It is used for keeping records of all the composite Web services available to registered users. Access count of composite services for each user is maintained. This count is used to list composite services in the descending order of frequency of composite services access. A data base called as MergOntoBase is maintained which contains OWL-S descriptions of Web services. One OWL-S description file is created for each composite service. It is generated by merging ontologies of Web services participating in a particular service composition. The core part of the framework is Web Service Composer. Its task is to carry out composition of Web services dynamically. For each precomposed service, whenever request comes, Web service composer uses OWL-S descriptions of Web services, completes composition process and responds to user request. If the request for composition is come for first time, copies of corresponding ontologies are used to generate single merged ontology for all participating Web services. MergOntoBase is updated accordingly. Details of ontology merging process are mentioned in the following section.

4. Modeling ontology merging using petri nets

Petri Nets allow us to model and analyse processes [15]. It provides us graphical notations for stepwise processes. We take advantage of this feature of Petri Nets to define workflow for merging Process Model ontologies. Nodes in Petri Nets represent transitions and places. We consider following features of Petri Nets useful for modeling ontology merging:

- Natural representation of change
- Natural representation of concurrency
- Mapping to other logics of change
- Execution semantics

In this section, we describe how Petri Nets are useful for ontology merging process before adding it to MergOntoBase.

Consider the example of virtual tourism system. We consider following Web services provided by tourism services:

- Travel Destinations Service (TDS) which provides list of different travel destinations in the selected country,
- Flight Booking Service (FBS)
- Hotel Booking Service (HBS)
- Online Payment System (OPS)

![Diagram of virtual tourism system]

Figure 2 shows partial OWL-S Process Models consisting of atomic processes for the provided Web services. The Process Model describes a service as a composite process. A composite process consists of composite processes and atomic processes. For TDS when a country name is given as input for process, it outputs a list of travel destinations in that country. User selects one of the travel destinations and searches for flight and hotel availability with help of HBS and FBS. After selecting convenient flight and hotel, user can confirm tour by paying online using OPS. We provide all these services in our framework as one composite service. While creating composite Web service, we Merge all these OWL-S descriptions in one document and add it to
MergOntoBase. It is important to decide sequence of these descriptions while merging.

The time required to parse MergOntoBase, when consecutive requests come for the composite Web service. We propose the use of Petri Nets decide the sequence of OWL-S descriptions. We need to consider relations between different atomic processes before proceeding for merge. With Petri Net dependency between atomic processes becomes clear, as we can show it graphically.

```
<process:AtomicProcess rdf:about="#RequestTravelLocation">
  <process:Input rdf:ID="CountryName"/>
  <process:ParameterType rdf:resource="#&xsd#string"/>
  </process:Input>
  <process:UnConditionalOutput rdf:ID="Locations"/>
  <process:ParameterType rdf:resource="Locations"/>
  </process:UnConditionalOutput>
  <process:hasOutput/>
</process:AtomicProcess>

(a)
```

```
<process:AtomicProcess rdf:about="#RequestForHotel">
  <process:Input rdf:ID="Period"/>
  <process:ParameterType rdf:resource="#Period"/>
  <process:Input rdf:ID="Location"/>
  <process:ParameterType rdf:resource="#Location"/>
  <process:ConditionalOutput rdf:ID="Rate"/>
  <process:ParameterType rdf:resource="#Rate"/>
  <process:ConditionalOutput rdf:ID="HotelName"/>
  <process:ParameterType rdf:resource="#&xsd#string"/>
  </process:AtomicProcess>

(b)
```

```
<process:AtomicProcess rdf:about="#RequestForFlight">
  <process:Input rdf:ID="Period"/>
  <process:ParameterType rdf:resource="#Period"/>
  <process:Input rdf:ID="Location"/>
  <process:ParameterType rdf:resource="#Location"/>
  <process:ConditionalOutput rdf:ID="Rate"/>
  <process:ParameterType rdf:resource="#Rate"/>
  <process:ConditionalOutput rdf:ID="Flight"/>
  <process:ParameterType rdf:resource="#&xsd#string"/>
  </process:AtomicProcess>

(c)
```

```
<process:AtomicProcess rdf:about="#BookTrip">
  <process:Input rdf:ID="Amount"/>
  <process:ParameterType rdf:resource="#Amount"/>
  <process:Input rdf:ID="PaymentDetails"/>
  <process:ParameterType rdf:resource="#PaymentDetails"/>
  <process:ConditionalOutput rdf:ID="TransactionId"/>
  <process:ParameterType rdf:resource="#&xsd#string"/>
  </process:ConditionalOutput>
  <process:hasOutput/>
</process: AtomicProcess>

(d)
```

Fig. 3 Partial OWL-S Process Models for
(a) TDS (b) HBS (c) FBS (d) OPS

In our example, travel location selected from TDS is used as input for HBS and FBS. The total amount for payment used by OPS is the sum of amounts for Hotel and flight booking which are outputs of HBS and FBS respectively.

In case when number of processes included in Web services composition is very large, these dependencies can be understood easily with help of Petri Nets.

**Definition (PM Net)**

A Process Model net is a tuple \( PMN = (P, T, F, I, O) \) where:
- \( P \) is a finite set of places
- \( T \) is a finite set of transitions
- \( F \subseteq (P \times T) \cup (T \times P) \) is a set of directed arcs (flow relation)
- \( I \) is a finite set of input place
- \( O \) is a finite set of output place

![Petri Net for OWL-S Process Models](image)

Fig. 4 Petri Nets for OWL-S Process Models for
(a) TDS (b) HBS (c) FBS (d) OPS
Generally, the notations \( \bullet t, \bullet t' \) indicate the sets of input and output places for transition \( t \) and \( t' \); \( p, \bullet p \) indicate the sets of transitions for which \( p \) is an input and output place. We map atomic operations mentioned in the Process Models to transactions in PM net. Things and classes in OWL-S are represented as places. The inputs and outputs of atomic processes map to transitions in PM net. We consider both control flow and data flow sequence of processes in OWL-S while modeling Petri Nets.

Figure 4 shows PM net for the Process Models shown in Figure 3. We use the bottom-up approach for modeling Petri Nets. We first construct PM Nets for component processes. We then generate the final net by merging and/or linking all the subnets. While preparing Petri Nets for composite processes, we need to model control constructs like sequence, if-then-else, split, split-join, any-order, repeat-while and repeat-until. These constructs define sequential execution, conditional execution or parallel execution of processes. It helps us to define sequence of processes to be added in the merged document.

![Diagram](image)

Fig. 5 An integrated Petri Net model

In Figure 5 we show an integrated Petri Net. It is generated by sharing the common places, transitions and data paths. We use this Petri Net model for deciding sequence of atomic processes to be added in the merged ontology. We can also understand from the diagram that, which processes can be added in any sequence as they can be executed parallel. Verification of ontology merging process before starting it is very important. With PM Net model we can find out differences, inconsistencies and redundancies between different OWL-S ontologies participating in merging process. These representations can also be used for analysing and verifying Web service composition process.

After completing integrated PM net, actual merging of process ontologies is carried out for Virtual Tourism Portal. While merging ontologies, we have verified which terms from ontologies need to be renamed, whether it is required to define new terms and also requirement of redefining of some existing terms. We checked whether an axiom is redundant and hence needs to be deleted from merged ontology. We consider both syntactic difference and semantic difference between ontologies. In syntactic comparison we compare a set of axioms. We also import declarations and annotations of ontologies. In semantic comparison, we find out logical differences between processes.

5. Implementation details

All Web services under virtual tourism system are developed using concepts of semantic Web. For each Web service description is provided in OWL-S language. We used java language to build the whole system. The OWL-S descriptions of Web services are built using Protégé Ontology Editor. The Web services composition is carried out with the help of APIs provided by MINDSWAP.

6. Impact analysis

We deployed virtual tourism system to find out effectiveness of our framework. Along with it we also used 3 more systems. First is Learning Resource Library which provides access to learning resources like eBooks, slides, videos and tutorials. Second system is Online Flower Shop. And last system we developed is Online Cricket Update system.

We prepared merged ontologies in OWL-S format for all four services both using Petri Net based approach and without using Petri Net based approach. While implementing the Precomposed services framework we found that error rate for merged ontologies prepared using Petri Net based model is much less than for those prepared without using Petri Net based model.

Table 1 shows precision values calculated by considering number of errors occurred while using merged ontologies without using our approach and using our approach.
We also calculated effectiveness our framework by considering customer’s response. We provided access to all these precomposed services from one portal and provided one more portal where access to individual Web services is given.

The individual Web services are those using which precomposed services are prepared. We kept track of number of users accessing these services for one week. Graph in Figure 6 shows effectiveness of our approach. Number of users accessing precomposed services framework is more than the individual Web service access.

7. Conclusion

We have described a novel approach to carry out ontology merging using Petri Nets while carrying out web service composition. With the example of virtual travel guide we explained how to represent OWL-S Process Models as Petri Nets. After building integrated Petri Net, we defined the sequence of inputs i.e. atomic processes for creating merging ontology. We have shown that the time required to parse such ontology is less. The approach can also used to verify Web service composition and find out degree of parallelism that can be achieved while executing Web service composition procedure.

Table 1. Improvement in precision values using our approach

<table>
<thead>
<tr>
<th>System</th>
<th>Precision using our approach</th>
<th>Precision without using our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Tourism</td>
<td>0.758403</td>
<td>0.734819</td>
</tr>
<tr>
<td>Learning resources library</td>
<td>0.767647</td>
<td>0.74623</td>
</tr>
<tr>
<td>Online Flower Shop</td>
<td>0.80925</td>
<td>0.73364</td>
</tr>
<tr>
<td>Online Cricket Update</td>
<td>0.814851</td>
<td>0.766474</td>
</tr>
</tbody>
</table>

References

[4] Ping Ren, Yajuan Song, Lei Liu. Web Service Composition Based on the Annotated Ontology, in Fifth International workshop on education technology and computer science, IEEE Xplore, 2009