Discovery of Semantic Relations between Web Services

Lin Lin & I. Budak Arpinar
LSDIS (Large Scale Distributed Information Systems) Lab
Department of Computer Science, University of Georgia
Athens GA 30602-7404, {lin, budak}@cs.uga.edu

Abstract

Discovering and assembling individual Web services into more complex yet new and more useful Web processes has received significant attention from academia recently. In this paper, we explore using pre and post-conditions of Web services to enable their automatic composition. Also, we present a novel technique for discovering semantic relations between pre and post-conditions of different services using their ontological descriptions. This enables determining services with complementary functions and generating a semantic Web of services. Our technique takes semantic similarity of pre and post-conditions into account and builds on our earlier work on discovering semantic relationships between interfaces (input and output) of Web services. A comprehensive classification of existing composition techniques is also included.

1. Introduction

Web services (WSs) extend the current Web from a distributed source of information to a distributed source of services. They are designed to provide interoperability between diverse applications, i.e., the platform and language independent interfaces of WSs allow the easy integration of heterogeneous applications. For example, the languages such as Universal Description, Discovery, and Integration, Web Services Description Language and Simple Object Access Protocol define standards for service discovery, description and messaging protocols.

The semantic Web is also an extension of the current Web in which information is given well-defined meaning, consequently better enabling computer and human to work in cooperation. Semantic Web aims to add machine-interpretable information to Web content in order to provide intelligent access to heterogeneous and distributed information. In a similar way, ontological concepts are used to define semantic Web services i.e., services supporting automatic discovery, composition, invocation and interoperation. As part of the DARPA Agent Markup Language program, OWL-S, an ontology of services is developed as a set of language features arranged in those ontologies to establish a framework within which the Web services may be described in this semantic Web context.

Developing efficient automatic discovery and composition techniques is among the most important challenges for the success of semantic Web services. Finding a suitable way to put these two features together has become one of the key points to convert the Web into a distributed source of computation, as they enable the location and combination of distributed services to provide a required functionality. To contribute towards this goal, we developed an Interface-Matching Automatic (IMA) composition technique earlier [1 & 2]. The possible compositions are obtained by checking semantic similarities between interfaces of individual services without any predefined template and user's involvement in specification and adaptation. An optimum composition which can best satisfy a user's needs considering the semantic similarity and quality is selected. However, our experiments show that without functionality constraints, IMA technique is more appropriate for the information-retrieval services, which always return relatively simple results based on the user-supplied inputs [1]. This is mainly due to the fact that WSs with the same interface could have different functions and the difficulty of mapping of input and output parameters for many services.

In order to overcome the problems we encountered in our previous work [9 & 10], we propose a discovery technique based on pre and post-conditions of WSs. We believe that the pre and post-conditions can semantically express the capabilities of services in a simple manner if they are expressed as a set of RDF (Resource Description Framework) triples. In this paper, we present a novel technique to identify possible relations between pairs of WSs by checking semantic similarities between their pre and post-conditions. Using an ontology, we can discover

relations between two services even the conditions don't match each other syntactically. This technique also addresses the issue of relaxed matching in the sense that pre-condition of one service can be satisfied by two or more WSs.

In particular, our work targets the following problem: given a set of WSs, the semantic relations between pre and post-conditions of these services need to be established, and then a semantic network of services with complimentary functions can be constructed according to these relations. Therefore, the subsequent step of composition according to a specific task at hand can be viewed as a path traversal problem. An extended version of this work can be found in [9].

The rest of the paper is organized as follows: Section 2 reviews the related work on WS composition techniques. Section 3 reviews our earlier work on IMA technique. Section 4 presents the technique for discovering semantic relations between pre and post-conditions of WSs. Section 5 describes the system architecture and experimental results, and finally Section 6 provides the conclusions.

2. Related Work

2.1 Template-based Techniques

Template-based techniques compose an application from a given service template. A service template defines types or rules of the components required for composing an application, as well as structure of the application. A user can choose a service template from a repository or create it him/herself. An adaptation of these template-based techniques, called process-driven techniques, is emerging as a promising approach to integrate business applications within and across organizational boundaries. In this approach, individual WSs are federated into composite WSs whose business logic is expressed as a process model. This process model identifies the functionalities required by the services to be composed (i.e., the tasks of the composite service) and their interactions (e.g., control and data flow, and transactional dependencies). Component services that are able to provide the required functionalities are then associated with the individual tasks of the composite services and invoked during each execution of the composite service. In eFlow [6], the process is modeled as a graph that defines the control and data flow. MWSCF [16] captures semantics of the activities in the process. The activities are not bound to WS implementations but defined using semantic descriptions.

The adaptability of the template-based systems is limited since they cannot compose the applications whose templates are not available. Also, creating service templates requires technical knowledge and experience. In addition, many of the existing template-based systems, such as eFlow [6], adopt a centralized architecture where centralized servers store and process service templates.

2.2 Interface-based Techniques

Interface-based techniques use interface information of WSs (i.e., a set of inputs and outputs) instead of service templates in order to compose an application. A user requests an application by submitting interface information of the application s/he needs. The requested application is composed through combining services such that the combination of the services accepts the requested inputs and generates the requested outputs.

The interface-based techniques have limited flexibility since certain services cannot be represented through a set or inputs and outputs. For example, a service that sends a text message to a specified email address cannot be modeled as a set of inputs and outputs since it does not output any data. Also, services may have more than one input and output parameters, and their interfaces may not match syntactically. [1] addressed this issue by proposing ontology-driven techniques, which will be discussed in Section 3. Furthermore, interface information usually provides little or no semantic information about the internal functionality of WSs.

2.3 Logic-based Techniques

Logic-based techniques extend the interface-based approach by usually adding first-order formula as pre and post-conditions into interface information. A user requests an application by submitting a formula representing the logic that must be satisfied by the application. The requested application is composed through combining components such that the conjunction of the logics specified in the components is equivalent to the logic specified by the user. SWORD [13] and SHOP2 [18] follow this approach.

2.4 Ontology-driven Techniques

Ontology-driven techniques extend the interface-based approach by bridging the concept gaps in interface parameters and other parts of the descriptions of services [17]. For example, MWSAF (METEOR-S

Web Services Annotation Framework) was designed to annotate WSDL files with relevant ontologies [12]. MWSCF (METEOR-S Web Services Composition Framework) makes use of ontologies in template definition to allow much richer description of activity requirements [16].

The use of ontologies for matching interface parameters leads to different degree of similarity matches. For example, if the output parameter of the former service subsumes the input parameter of the succeeding service, the properties of the parameters could be partially satisfied. Thus, this kind of weak matches may not always guarantee the correctness of the composition.

2.5 Quality-driven Techniques

The quality-driven techniques extend the process-driven techniques by adding the selections of component services based on a set of quality criteria during execution of a composite service. The number of services providing a given functionality, although with different levels of pricing and quality, may be large and constantly changing. Consequently, it may be inappropriate to compose composite services that require the identification of the exact services at the design-time. A WS can be selected during execution based on some operational metrics of non-functional properties, such as reliability, execution price, duration, reputation, availability, and security. The runtime selection of component services during the execution of a composite service has been put forward as an approach to address this issue. For example, [5] presents a QoS model for workflow components.

2.6 Automata-based Techniques

The automata-based techniques use Finite State Automata (FSA) to model a WS composition. FSA is widely known as a simple but powerful formalism, which allows to model the behavior of a system as a sequence of transitions. In [4] a service is modeled as a Mealy machine, with input and output messages, and a queue is used to buffer messages that were received but not yet processed. [3] describes a composition model involving activity-focused FSA. One input to this approach is a set of descriptions of component WSs, each given as an automaton. The second input is a desired global behavior, also specified as an automaton, which describes the possible sequences of activities. The output is a subset of the component services, and a mediator.

2.7 Petri net-based Techniques

These techniques use Petri nets to model processes. Petri nets are well founded process modeling technique that has formal semantics. After a Petri net is defined for each service, composition operator, such as sequence, selection and iteration, are used to perform composition. For example, [11] encodes WS descriptions in a Petri net formalism and provides decision procedures for WS simulation, verification and composition. [7] proposes a Petri net-based algebra for composing Web services. The formal semantics of the composition operators is expressed in terms of Petri nets by providing a direct mapping from each operator to a Petri net construction. Thus, any service expressed using the algebra constructs can be translated into a Petri net representation. A colored Petri net-based approach proposed in [19] captures both complex conversation protocols and process compositions.

3. Semantic Relations between Inputs and Outputs

Our earlier work investigated the semantic relations between inputs and outputs of WSs by checking their semantic similarities [1 & 2]. We developed an Interface-Matching Automatic (IMA) composition technique that aims for generation of complex WS compositions automatically. This requires capturing user's goals (i.e., expected outcomes), and constraints, and matching them with the best possible composition of existing services. Therefore, inputs and outputs of the composite service should match the user-supplied inputs, and expected outputs, respectively. Furthermore, the individual services placed earlier in the composition should supply appropriate outputs to the following services in an orchestrated way similar to an assembly line in a factory so they can accomplish the user's goals.

In IMA, we navigate the process ontology to find the sequences starting from the user's input parameters and go forward by chaining services until they deliver the user's expected output parameters. The composition terminates when a set of WSs that matches all expected output parameters is found, or the system fails to generate such a composition of services.

The goal of this technique is to find a composition that produces the desired outputs within shortest execution time and better data-flow (i.e., better matching of input and output parameters). Note that input parameters may not match syntactically yet they can be semantically equivalent. The degree of semantic

match is calculated using a function of quality rate and semantic similarity value.

However, our experiments show that sometimes IMA technique can fail to produce correct compositions due to the fact that some WSs, with same input and output parameters, provide quite different functionalities. Thus we have developed a (Human-Assisted Automatic) HAA composition technique to help users in selecting appropriate WSs among a ranked list, and build a composition incrementally [2]. We believe exploiting semantic relations between pre and post-conditions can further enhance quality and usability of produced compositions,

4. Semantic Relations between Pre and Post-conditions

The problem of composing autonomous WSs automatically to achieve new functionality is generating considerable interest in recent years in academia and industry as mentioned earlier. Automatic services composition requires an approach based on semantic descriptions, as the required functionality has to be expressed in a high-level and abstract way to enable reasoning procedures.

The required high-level functionality description can be viewed as the capability of the service. However, different services can provide the same capability (e.g., booking a flight) and the same service can provide different capabilities (e.g., searching a book or a movie through the same service). In this sense, capabilities must be naturally described separately from specific service descriptions so that generic functionalities can be expressed [8]. For example, several services offering the same functionality but with different inputs and outputs should be related to the same generic high-level capability. However, several services having same inputs and outputs but offering different functionalities should be distinguished from each other.

In order to compose services based on functionalities, there has to be a way to express the functionality of a service. Since pre and post-conditions can be used to define the capability of the service in terms of the information needed to use the service and the results of its invocation, functionalities of services can be expressed in terms of these conditions. So the problem of investigating the degree of interoperability of WSs based on semantic relations of their pre- and post-conditions becomes very important during WS composition.

4.1 Relationships among Web Services

We define two WSs to have a relationship as either these two services can be somehow plugged together to perform a valued added service or one of service can be substituted by the other. Let service S_m and service S_n be two services, and a relationship R between services S_m and S_n can be identified as follows:

- *Prerequisite Relationship*: ($S_m \rightarrow S_n$) The prerequisite relationship means that one service has to finish before the other starts. Service S_m has to finish before service S_n starts. For example, the booking service has to be done before the payment service.
- *Parallel Relationship*: (S_m // S_n) Here services S_m and S_n can execute in parallel but the results of each service need to be combined to enable further execution.
- *Substitute Relationship*: ($S_m \Leftrightarrow S_n$) Here service S_m can be substituted by service S_n. The services S_m and S_n seem to provide the same functionality but they have different attributes. For example, in the case of delivery service, service S_m can be an air courier delivery service while service S_n is a ground delivery service.
- *Include Relationship*: (S_m Ǝ S_n) The include relationship means that one service provides services that includes the services offered by the other. The service S_m includes the service S_n. For example, service S_m can be an express delivery service that offers both ground and air delivery while service S_n is a ground delivery service.

4.2 Modeling Pre and Post-conditions

The problem of determining the relationship between two services can be addressed through discovering semantic relations between the pre and post-conditions of these services using ontologies. Thus, finding a suitable way to express pre and post-conditions semantically becomes a very important issue. It should be simple and expressive enough for machine processing, and capturing the functionality of services respectively.

Pre-conditions can be expressed as, but not limited to, high-level inputs to the service together with conditions over these inputs. High-level input means that more specific concepts in the ontology can be found to replace more abstract concepts in the input (e.g., indicating payment information as a pre-condition, instead of credit card information or bank information). It is important to notice that the pre-conditions of a service are not independent of each

other, as they all define the functionality. Post-conditions can also be expressed as, but not limited to, high-level results of the service execution together with conditions over these results. As with pre-condition, they cannot be considered independent, as the removal of one of them changes the functionality.

There is no strong consensus for representing pre and post-conditions in a certain specification language. Hence, we model pre and post-conditions of a WS as two sets of RDF triples from an ontology. The use of RDF triples provides a way that is simple yet rich enough to express pre and post-conditions semantically. For example, the pre-condition for a course registration service could be expressed as a set of triples: (*course status available, course has prerequisite, student pass prerequisite*), and the post-condition could be expressed as another set of triples: (*student register course*). Both pre and post-conditions in this simplistic example are defined at the schema level of the ontology. For example, the post-condition expresses that the student should register the course if the pre-condition is satisfied.

Note that how pre and post-conditions can be specified or if they can be automatically generated are out of scope of our work.

4.3 Semantic Relations between Conditions

Each set of RDF triples in a pre or post-condition is simply called a condition. Thus, a WS can be viewed as Cond*1*→Cond*2*, in which Cond*1* and Cond*2* represent the pre and post-condition of the WS respectively. The relationship between two services can be identified by checking the semantic relations between these conditions of WSs.

Let service S*m* and service S*n* be two services, and service S*m* can be represented graphically as *Condm1*→*Condm2* in Figure 1(a), and service S*n* can also be represented graphically as *Condn1*→*Condn2* in Figure 1(b). Several semantic relations between conditions can be identified as follows:

- Condition Cond*m2* exactly matches to condition Cond*n1*: *Condm2* ➔ *Condn1*. In Figure 1(c), the exact match relation between conditions Cond*m2* and Cond*n1* is represented graphically, and services S*m* and S*n* have the prerequisite relationship (e.g., →).
- Condition Cond*m2* is semantically stricter than condition Cond*n1*. Figure 1(d) shows that Cond*m1* can be a plug-in (PI) match to Cond*n1*: *Condm2* PI➔ *Condn1*. In this case, services S*m* and S*n* have the prerequisite relationship. For example, condition Cond*m2* can specify the availability of payment by MasterCard only, and condition Cond*n1* can allow the availability of payment by all major credit cards.

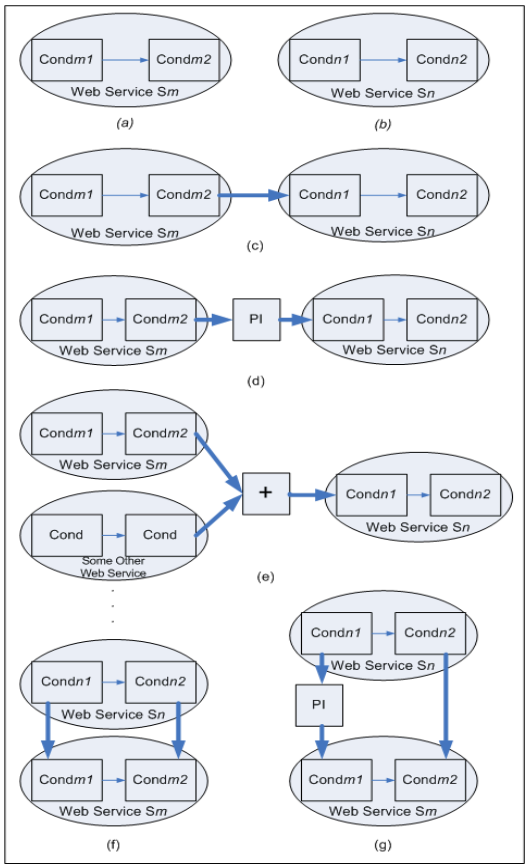

Figure 1: Semantic Relations between Conditions

- Condition Cond*m2* only partially satisfies condition Cond*n1* so that some other condition(s) are needed together with Cond*m2* to completely satisfy Cond*n1*. We say that Cond*m2* can be a plus-match to Cond*n1* as shown in Figure 1(e): *Condm2* +➔*Condn1*. Service S*m* has the parallel relationship (i.e., //) with some other services since their results together enable the execution of service S*n*.
- Condition Cond*m2* compliments condition Cond*n1*. For example, condition Cond*m2* can specify that book is available to be sold, and condition Cond*n1* can specify that book is available to be bought. Buying denotes an action that is compatible to selling. While [14] defines this compatible relation in their action-resource ontology, we can define this kind of relations in condition ontology. We say that condition Cond*m2* can be a complimentary match to condition Cond*n1*: *Condm2* CP➔

Condn1. In this case, services S*m* and S*n* also have the prerequisite relationship.

Figure 1(f) shows that services S*m* and S*n* have the substitute relationship (i.e., ⇔) since condition Cond*n1* exactly matches condition Cond*m1*, and condition Cond*n2* exactly matches condition Cond*m2*. Figure 1(g) shows the include relationship (i.e., Э) between services S*m* and S*n*. In this case, condition Condn1 is a plug-in match to condition Cond*m1* while condition Cond*n2* exactly matches to condition Cond*m2*. For example, as mentioned earlier, Cond*m1* can specify that express delivery, including air and ground deliveries, is available while Cond*n1* expresses that only ground delivery is available. Post-conditions of two services Cond*m2* and Cond*n2* can be the same as they both express the effect of executing a delivery service.

4.4 Discovery of Semantic Relations between Conditions

The degree of similarity between two conditions is assessed through comparing similarity between triples of these two conditions. To evaluate the similarity of two triples, each component from the triples are compared, and a similarity value is assigned. In the following, we explain the relation discovery algorithm in several steps:

1. evaluating the similarity of two triples,
2. calculating the similarity value between two conditions,
3. identifying the semantic relation between two conditions using similarity value, and
4. identifying the semantic relations between pre and post-conditions among services.

The first step is to evaluate the similarity of two triples. We consider the following cases of similarity measure for a pair of triples:

1. If all three corresponding components between two triples are same, the similarity value for these two triples is maximal.
2. If one or more components from the first triple are subsumed by the corresponding component(s) from the second triple, their similarity value is the second best. For example, the two triples, *book hasAuthor person* and *mediaObject hasAuthor agent*, fall into this case since *book* from the first triple is subsumed by *mediaObject* from the second triple, and *person* from the first triple is also subsumed by *agent* from the second triple.
3. If one or more components from the first triple subsume the corresponding component(s) from the second triple, their similarity value is the third best.
4. If at least one component from first triple is subsumed by the corresponding component from second triple (case 2) while the other one or more components from first triple subsume the corresponding component(s) from second triple (case 3), their similarity value is the fourth best.
5. If they have no subsumption relation, the similarity value can be obtained by using Tversky's feature-based similarity model [5]. Usually, in this case, their similarity value is very small.

We use a linear combination of the similarity values of each triple in the condition. Currently we manually set the threshold value that distinguishes "good" matches from "bad" matches. A triple with similarity value greater than the threshold value is considered as a good-matched triple, otherwise it is considered as a bad-matched triple. In our experiments, the threshold value is set to be the fourth best similarity value described in case 4 in the previous step so that a triple with case 4 or case 5 similarity value would be considered as a "bad" match. The reason for the choice of the threshold value is that this high threshold value will guarantee the correct semantic relations being identified in the following steps. Thus, for a pair of conditions, the similarity values of good-matched triples are combined linearly. To further obtain the normalized similarity value for two conditions, the similarity value is divided by the total number of good-matched triples in the conditions.

The next step is to map this normalized similarity value to one of three semantic relations between conditions identified in the previous section or to ignore it because the value is too small to be considered.

The final step is to find the semantic relations between pre and post-conditions among a set of Web services. For every pair of services, their pre and post-conditions need to be cross matched to obtain all possible relations. Suppose service S1 has pre-condition Pre1 and post-condition Post1, and service S2 has pre-condition Pre2 and post-condition Post2. Then, the relations need to be identified between four pair of conditions: *(i)* Pre1 and Pre2, *(ii)* Post1 and Post2, *(iii)* Post1 and Pre2, and *(iv)* Post2 and Pre1. Here, identifying relations in *(i)* and *(ii)* may discover possible substitute and include relations between two services; and possible prerequisite and parallel relations between two services can be discovered by

identifying relations in *(iii)* and *(iv)*. According to the identified relations between pre and post-conditions, a semantic network of WSs can be generated.

It must be noted that we only address this issue for services having only a single pre-condition and a single post-condition. The services having multiple pre and post-conditions are out of scope of this paper.

6. System Architecture and Experiments

For the experimental purposes, we have developed a prototypical system (Fig. 2). The Reasoning Engine takes the pool of WSs as input, and identifies the semantic relations between the pre and post-conditions of each pair of WSs. The Reasoning Engine sends the RDF triples from pre and post-conditions of services to the Matching Engine for identifying similarity values. This engine uses Jena APIs to load and query the ontology to locate relations between two given concepts. The resulting semantic Web of services is visualized through a Touch Graph based graphical user interface [9].

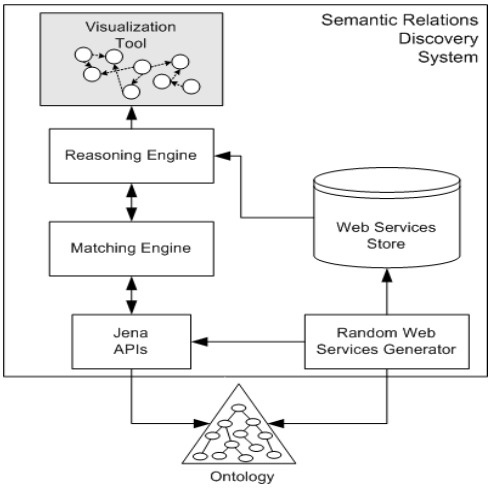

Figure 2: An Overview of System Architecture

Due to the lack of the standard semantic WSs dataset we have created a collection of synthetic WSs using TAP, which is an experimental knowledge base about people, places, products, etc [15]. We evaluated our discovery results with respect to those obtained by a panel of ten human subjects, who are graduate students in computer science and not familiar with the research presented here. The human subjects were given thirty randomly generated services, each consisting of pre and post-condition. Together with the dataset, all subjects were provided with the discovery criteria for three types of relations between conditions. They were also provided with a graphical representation of partial domain ontology used to generate pre and post-conditions for the service dataset, thus allowing them to evaluate the similarity between triples within conditions. They then identified all possible relations between pre and post-conditions of all thirty services according to the criteria provided.

In order to demonstrate the effectiveness of our discovery scheme, we illustrate in Fig. 3 the performances of human subjects and system. The x-axis represents ten human objects and system, each having three columns indicating its performances on exact match, plug-in match, and plus match relations, respectively. On the other hand, the y-axis represents the number of correct relations identified by the system and human subjects. It is evident in the figure that there are varying levels of performances in human subjects' ability to identify semantic relations between conditions. Especially, they had difficulty of identifying plus match relation. Note that the system identifies more relations than those human subjects. Fig. 4 shows the performances of the system with different threshold values. The average performance is also calculated and displayed in Fig. 4. It is a clear indication that varying threshold values can produce varying levels of performances. To obtain a good system performance, we recommend that the threshold value is set to the value previously discussed.

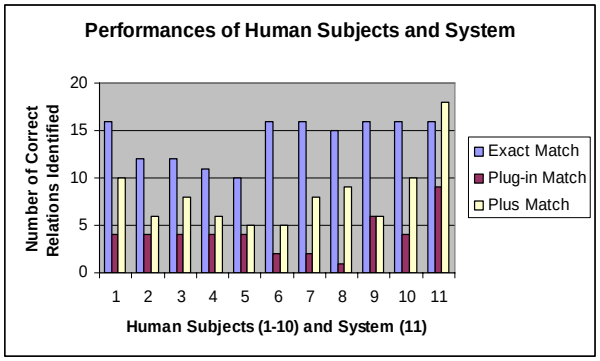

Figure 3: Performances of Human Subjects and System

Our experiments are based on a synthetic dataset of services with only pre and post-condition specifications and numerical identifiers. It is not surprising to see that the performance of the system is better than those of human subjects in the experiments since they can not apply their real world experiences to a synthetic dataset that has no real meanings. If real services, with a full description of service name, goal, and i/o parameters as well as pre and post-conditions, were to be used in the experiments, human subjects might perform better than the system. Human subjects are good at identifying relations, for example, using

WS names, when various sources of information are provided. However, there is a limit for the amount of WS specifications human subjects can handle at a time. As the number of services in the dataset increases significantly, it becomes difficult for them to identify all possible relations among pre and post-conditions. In this case, our technique will give a better performance with large number of services in a dataset. Also, frequently WS names may not be descriptive enough so that human subjects may have difficulties in identifying possible relations in this case as well.

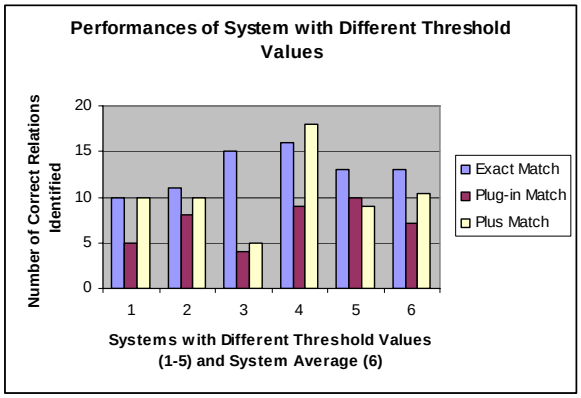

Figure 4: Performance with 5 Different Thresholds

7. Conclusion and Future Work

In this paper, we propose a novel technique for discovering semantic relations between pre and post-conditions of different services using their ontological descriptions. Currently, we didn't test the actual compositions in our prototype. However, users can visually browse the generated semantic Web of pre and post-conditions of services for checking possible compositions. Also path traversal algorithms can be applied to our semantic Web of services to obtain service compositions. Our future work also includes conducting more experiments when standard semantic WSs datasets become available.

8. References

[1] I. B. Arpinar, R. Zhang, B. Aleman-Meza, and A. Maduko., "Ontology-driven Web Services Composition Platform", *IEEE Intl. Conf. on e-Commerce Technology*, San Diego, California, July 6-9, 2004.

[2] I. B. Arpinar, R. Zhang, B. Aleman-Meza, and A. Maduko, "Ontology-driven Web Services Composition Platform", *Journal of Information Systems and e-Business Management*, 3(2):175-199, July 2005.

[3] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, "Automatic Composition of E-services that Export Their Behavior", *Proc. of the 1st Intl. Conference on Service Oriented Computing*, 2003.

[4] T. Bultan, X. Fu, R. Hull, and J. Su, "Conversation Specification: A New Approach to Design and Analysis of E-Service Composition", *Proc. of WWW Conference*, 2003.

[5] J. Cardoso, *Quality of Service and Semantic Composition of Workflows*, Ph.D. Dissertation, Dept. of Computer Science, Univ. of Georgia, Athens, GA, 2002.

[6] F. Casati, S. Ilnicki, L.-J. Jin, V. Krishnamoorthy, and M.-C. Shan, "Adaptive and Dynamic Service Composition in eFlow", *Proc. of the Intl. Conf. on Adv. Info. Systems Engineering*, Sweden, 2000.

[7] R. Hamadi and B. Benatallah, "A Petri net-based Model for Web Service Composition", *Proc. of the 14th Australasian Database Conf.*, 2003.

[8] R. Lara, H. Lausen, S. Arroyo, J. de Bruijn, and D. Fensel, "Semantic Web Services: Description Requirements and Current Technologies", *Intl. Workshop on E-Commerce, Agents, and Semantic Web Services*, Pittsburgh, Sept. 2003.

[9] L. Lin, *Discovering Semantic Relations between Web Services Using Their Pre and Post-Conditions*, MS Thesis, Computer Science, University of Georgia, May 2005.

[10] L. Lin, and I. B. Arpinar, "Discovering Semantic Relations between Web Services Using Their Pre and Post-Conditions", *2005 IEEE International Conference on Services Computing*, July 2005, Orlando, Florida (poster).

[11] S. Narayanan and S. Mcllraith. Simulation, "Verification and Automated Composition of Web Services", *Proc. of the 11th Intl Conf. on WWW*, Hawaii, 2002.

[12] A. Patil, S. Oundhakar, A. Sheth, and K. Verma, "METEOR-S Web Service Annotation Framework", *Proceeding of the WWW Conference*, July 2004.

[13] S. R. Ponnekanti, and A. Fox, "SWORD: A Developer Toolkit for Web Service Composition", *Proc. of the 11th Intl Conf. on WWW*, Hawaii, 2002.

[14] M. Stollberg, U. Keller, and D. Fensel, *Partner and Service Discovery for Collaboration Establishment with Semantic Web Services*, Digital Enterprise Res. Inst., Univ. of Innsbruck, Austria, 2005.

[15] TAP KB, http://tap.stanford.edu/tap/tapkb.html

[16] K. Verma, *Configuration and Adaptation of Semantic Web Processes*, PhD Thesis, Computer Science, Univ. of Georgia, June 2006.

[17] WSDL-S, W3C Member Submission on Web Service Semantics, http://www.w3.org/Submission/WSDL-S/

[18] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau, "Automating DAML-S Web Services Composition Using SHOP2", *Proc. of 2nd Intl. Semantic Web Conference*, Sanibel Island, Florida, October 2003.

[19] X. Yi and K. Kochut, "Process Composition of Web Services with Complex Conversation Protocols: a Colored Petri Nets Based Approach", *Proc. of Design, Analysis, and Simulation of Dist. Sys. Symposium*, 2004.