Automatic Composition of Semantic Web Services*

Ruoyan Zhang, I. Budak Arpinar, and Boanerges Aleman-Meza
Large Scale Distributed Information Systems (LSDIS) Lab
Computer Science Department, University of Georgia
Athens, GA 30602-7404, budak@cs.uga.edu

Abstract

With the growing number of Web services, importance of composing existing Web Services into more complex services in order to achieve new and more useful solutions is increasing. However, in order to automatically compose new services, existing services need to be encoded in a machine understandable form. The semantics of a service can be described by annotating it with respect to service ontologies. The goals of automatic composition include reducing the complexity of creating composite services as well as choosing an optimal composition among possible options. This paper describes the Interface-Matching Automatic Composition technique that aims for generation of complex Web Services automatically by capturing user's expected outcomes when a set of inputs are provided; the result is a sequence of services whose combined execution achieves the user goals.

Keywords: Web Services, Composition, Semantics

1. Introduction

In recent years, a growing number of Web Services (WSs) have emerged as the Internet develops at a fast rate. The Web is now evolving into a distributed device of computation from a collection of information resources [Fensel02]. Furthermore, the need for composing existing WSs into more complex services is also increasing, mainly because new and more useful solutions can be achieved. However, the composition of discovered services and enabling data-flow among them are usually done manually, which are highly inconvenient, especially for more complex compositions.

Our service composition technique aims for reducing the complexity and time needed to generate, and execute a composition and improve its efficiency by selecting the best possible services available at the current time. In general, there are four different dimensions for a service composition: (i) degree of user involvement in a composition definition, (ii) if the composition is based on a template or actual service instances, (iii) dynamicity of the composition, and (iv) degree of user involvement in the dynamicity (or adaptation) of the composition. In an automatic composition, a user is not involved instead the system defines control and data-flow by assembling individual services. This is very challenging due to difficulty of mapping user needs to a collection of correlated services where their interim outputs can satisfy each other's input requirements and the final deliverable meets the user demands. Besides that, in each of these composition options either actual service instances or some generic templates are assembled. In the latter, individual services are searched and integrated automatically at execution time for a given plan [Chandrasekaran03].

In a dynamic composition (either user-defined or automatically-defined based on instances or templates), the composition itself can be adapted mainly because of quality of service (QoS) requirements at run-time by a user or automatically (i.e., user-adapted or automatically-adapted). Finally, a composition may not be defined at design-time but can be assembled service by service at execution time.

2. Related Work

A composition can be based on templates. An example is a trip planner, which is declared as a state chart, and the resulting composite services are executed by replacing the roles in the chart by selected individual services [Benatallah02]. The ICARIS project [Tosic01] and [Narayanan02] also use pattern composition approach. METEOR-S platform provides a comprehensive framework for semantic Web services and their composition [METEOR-S03]

The instance composition approach is to generate a composite service plan out of existing services. In this category, [Mao01] proposes a composition path, which is a sequence of operators that compute data, and connectors that provide data transport between operators. The search for possible operators to construct a sequence is based on the shortest path algorithm on the graph of operator space. However, [Mao01] only considered two kinds of services – operator and connector with one input and one output parameter (which is simplest case for service composition). Also in the instance composition category,

* This work is supported by University of Georgia Junior Faculty Grants.

SWORD uses a rule-based expert system to determine if a plan of composite service can be built out of existing services [Ponnekanti02]. It mainly focused on the composition of information provider services (i.e., not world-altering services), and (like [Mao01]) it does not address the input and output mismatch problem. In our approach, services can have more than one input and output, and these parameters can mismatch in the composition process.

3. Modeling Semantic Web Services

A Semantic WS is a unit of composition that can be deployed independently, and may be subject to composition by a third party on the Web. At the same time, its interface, its process specification (i.e., its functionality) and its relations to other services are defined, and advertised in a machine-processable form so it can be automatically discovered, composed, and invoked in new complex WSs. The emerging Semantic Web makes it possible to specify semantics of a domain such as the terms and concepts of interest, their meanings, relationships between them and the characteristics of the domain through an ontology. In this paper, a WSs ontology is used to define precise semantics of both individual, and complex service instances, as well as abstract services from which the properties, and process definitions are inherited.

3.1 Web Services Ontology and Service Profile

A WSs ontology describes the interfaces of the services and the relationships among them. An abstract service specifies names and types of input and output parameters with no property constraints. Like domain ontologies, a service inherits the properties of its parent service in a WSs ontology (see Figure 3.1 for price search ontology).

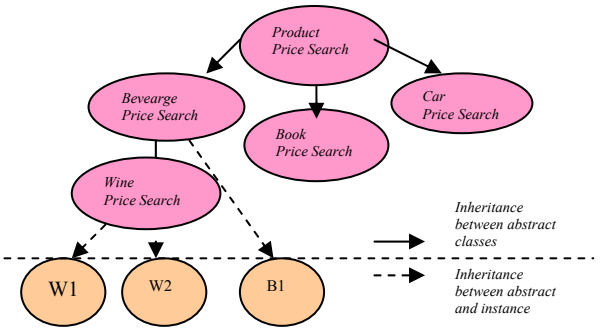

Figure 3.1: Price Search Ontology and Service Instances

In ONTOS we use emerging DAML-S service ontology [Ankolenkar02]. A service profile is the core element of a DAML-S specification, and it involves semantic descriptions of service interfaces and functions. In ONTOS, we primarily focus on the collection of inputs, and outputs for composition and process-oriented extension for functionality representation.

3.2 Query Format

A composite service query is represented in a very similar way as a service description in DAML-S (Figure 3.2). Like DAML-S template of services, the query profile includes the description of the composite service and the interface of the expected composite service, in which we define the output parameters, output constraints, input parameters, and their constraints. The output constraint specifies the requirements on the outputs by the user. The second part of the query is about the functionality of the composite service. The user can partially specify how the composite service works and what kind of individual services would be expected to be included (constraints and functionality parts are omitted in the figure for brevity).

Example: A restaurant owner wants to find matching wines to the meals in the restaurant and learn the prices of these wines.

Interface Query:
- Input Parameter 1: Food Type
- Output Parameter 1: Wine Prices
- Output Parameter2: Wine Name

Description Query:
- Find the matching wines to food
- Return the prices of the wines

Figure 3.2 A Composite Service Query

4. Interface-Matching Automatic Service Composition (IMA)

IMA composition technique 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 (i.e., pipe-and-filter) in a factory so they can accomplish the user's goals. Finally, the composition should conform to the user specified constraints including time, cost, and user specified quality of composition (QoC) properties.

In IMA, we navigate the WS 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 given the inputs provided by a user is found, or the system fails to generate such a composition of services.

The goal of this algorithm 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). If service ontologies are complex and the number of services is large this can be a challenging task. The composition starts from the service that needs one or more of the input parameters given by the user. If this WS does not produce all of the expected outputs, more WSs need to be found to provide the expected outputs. This process continues until we find a sequence of WSs that will produce the expected composition outputs from the user's inputs.

Figure 4.1 shows an extended WS ontology with new relations by matching input parameters and output parameters. Nodes represent services and edges connect services if the output of a service can be "feed-into" the input of a service. Edges shown with dash-lines represent parameters that are not exact match but they are semantically equivalent. In the figure, different service outputs can feed into other service inputs. For example service 6 requires two input parameters, one of which can be provided by either *S1* or *S3* and the other comes from *S4*.

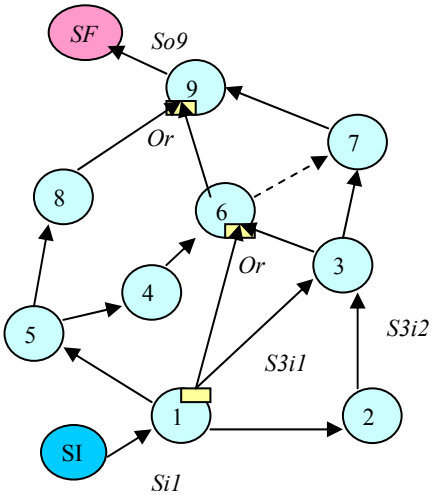

Figure 4.1: IMA Composition Technique

In an example scenario, the user provides input parameter *Si1* and expects the output *So9 as* indicated in the graph. The composition goal is to find a shortest sequence of services from *S1* to *S9*. In this graph the source node *SI* represents the start state and *SF* as the ending state, which are added for computing convenience. The weight of every edge is a function of execution time and semantic similarity value. As a matter of fact, other factors can be considered in computing weight of edges, such as reliability, security of services, etc. Relative weights of these factors (λ) are defined by the users as follows:

$$W = (\lambda) * \text{execution time} + (1-\lambda) * \text{similarity value}.$$

For the time being we consider four cases to check similarity (i.e., matching) of an output and input parameter from the same ontology: (1) if they are same, their similarity is maximal. For example, the output parameter of S1(in Fig.4.2) exact match with the input parameter of S3 and they have the smallest value 1.0 (2) If output parameter of the former service subsumes the input parameter of the succeeding service, this is the second best matching level, such as the output of S4 subsumes the expected output parameter - wine price. The similarity value depends on their distance in the ontology. (3) If the output parameter of the former service is subsumed by the input parameters of the succeeding service, the properties of the parameters could be partially satisfied. That applies to the relationship between S1 and S4. (4) When two parameters have no subsumption relation or they are from different ontologies, such as S2-S3, the similarity value can be obtained by using Tversky's feature-based similarity model [Cardoso02], which is based on the idea that common features increase the similarity of two concepts, while feature difference decreases the similarity.

The composition algorithm aims to find the optimal collections of services considering execution time and semantic matching of parameters. We modify Bellman-Ford shortest-path dynamic programming algorithm to find the shortest sequence from initial stage at node *SI* to the termination node *SF*. In a common directed graph, we consider only one incoming edge and one outgoing edge for every node selected in the shortest path. The difference in our graph representation is that some services need more than two incoming edges as input parameters. Therefore, we not only record distance for every node, but also we trace the distance of every path at every node. When all the required input parameters are available, a service can be executed. Therefore, the distance of every node is determined by the maximum value of distances of all the input parameters. For example, *S3* must have two incoming edges so a distance value of *S3* is determined by the maximum of *S3i1* and *S3i2* because *S3* can be executed after both of these inputs are available. In a different case, when there is more than one incoming edge fitting for one input parameter of a service, such as either edge 3-6 or 1-6 satisfies input of *S6*, we choose the minimum distance of 3-6 and 1-6 as a

distance associated with input parameter of *S6*. The algorithm running time is $O(n^3)$.

In Fig. 4.2, the user inputs the seafood and awaits the matching wine prices. Both Wine Agent (S1) and World Wine Agent (S2) are food-wine matching services which output the name of matching wines with food input by the users. Wine Price Information (S3) and Beverage Price Information (S4) provide the prices of corresponding wine or beverage. Edge weights are composed of two values representing the similarity degree value and annotating execution time of the precedent service. For example, the weight of edge <S2, S4> is 1.5 + 1, which means that similarity value of Wine:O3 (output of S2) and Beverage:O3 (input of S4) is 1.5 and S2 execution time is 1 unit (O3 means it is defined in ontology 3). Wine is subsumed by Beverage, therefore some information of beverage would be lost and the similarity value is larger than 1 If λ is set to different values by the user, we obtain different shortest paths in IMA technique (see Table 4.1).

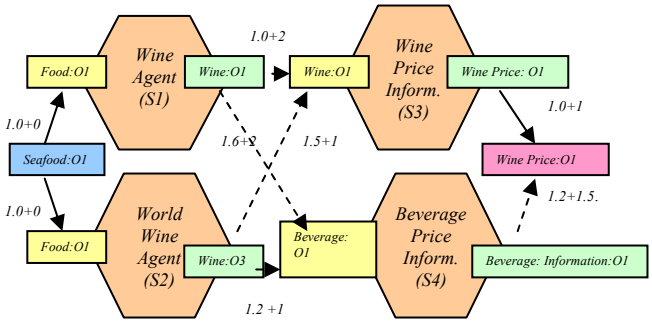

Figure 4.2 Wine Services Composition Example

λ	Shortest Path	Distance
λ=0	S1 -> S2	3
λ=1	S2 -> S4	2.5
λ=0.5	S1-> S3	2.6

Table 4.1 Shortest path under different λ

In more complex cases, there could be more than one input and output parameters for a composite service. In this case, our strategy is to compute the shortest path from every starting node to the possible destinations, then select the shortest path for every destination node

5. Conclusion

Automatic composition of Web services is a challenging research problem. Due to increasing number and heterogeneity of available Web services we rely on service semantics to automatically compose new services. Interface-Matching Automatic Composition technique incorporates the use of WS ontologies to find matching inputs and outputs. On the other hand, we are in the process of developing a Human-Assisted Automatic Composition technique that can complement the IMA composition technique through enabling human-involvement where composition can not proceed automatically or there are ambiguities in matching services.

Other directions in Web services composition that we are currently addressing include using functionality in order to compose complex services more efficiently.

6. References:

[Ankolenkar02] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, et. al. DAML-S: WS Description for the Semantic Web. The First Intl Semantic Web Conference (ISWC)

[Benatallah02] B. Benatallah, M. Dumas, Q. Z. Sheng, and A. H. Ngu. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. IEEE Intl. Conf. on Data Engineering, San Jose, California, Feb 2002.

[Cardoso02] J.Cardoso, and A.Sheth. Semantic e-Workflow Composition. Journal of Intel. Info. Systems.

[Chandrasekaran03] S. Chandrasekaran, J. A. Miller, G. Silver, I. B. Arpinar and A. P. Sheth, "Performance Analysis and Simulation of Composite WSs," Electronic Markets: The Intl Journal of Electronic Commerce and Business Media, Ronald Klueber and Heiko Ludwig (Guest Editors) Vol. 13, No. 2 (Spring 2003).

[Fensel02] D. Fensel, C. Bussel. Semantic Web Enabled WSs. In Second Annual Diffuse Conference:will WSs Revolutionize e-Commerce? Brussel,Belgium, January.

[Mao01] Z. M. Mao, E. R. Brewer, and R. H. Katz. Fault-tolerant, Scalable, Wide-Area Internet Service Composition. U.C. Berkeley Technical Report UCB//CSD-01-1129, Jan 2001.

[METEOR-S03] METEOR-S: semantic Web Services and Processes, lsdis.cs.uga.edu/proj/meteor/SWP.htm

[Narayanan02] S. Narayanan, and S. A. Mcllraith. Simulation, Verification and Automated Composition of Web Services. 11th Intl. World Wide Web Conference (WWW2002), Honolulu, 2002.

[Palucci02] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities", *The First Intl Semantic Web Conference*, Sardinia (Italy), June, 2002.

[Ponnekanti02] S. R. Ponnekanti, and A. Fox. SWORD: A Developer Toolkit for Building Composite Web Services. 11th WWW Conference, Honolulu, 2002.

[Tosic01] V. Tosic, D. Mennie, and B. Pagurek. On dynamic Service Composition and Its Applicability to E-business Software Systems. WOOBS'01 (Workshop on Object-Oriented Business Solutions) workshop (at ECOOP 2001), Budapest, Hungary, June 18, 2001.