

Web Service Composition using Service Suggestions

Rui Wang¹, Chaitanya Guttula¹, Maryam Panahiazar³, Haseeb Yousaf¹,
John A. Miller^{1,3}, Eileen T. Kraemer^{1,3} and Jessica C. Kissinger¹⁻⁴

¹Department of Computer Science

²Department of Genetics

³Institute of Bioinformatics

⁴Center for Tropical & Emerging Global Diseases

University of Georgia

Athens, GA, USA

{wang@cs., guttula@cs., mp@, haseeb@cs., jam@cs., jkissing@, eileen@cs.}uga.edu

Abstract— This paper presents a semi-automatic Web service composition approach. This approach ranks all available candidate Web service operations based on semantic annotations and suggests service operations to a human designer during the process of Web service composition. The ranking scores are based on data mediation, functionality and formal service specifications. A formal graph model, an IODAG, is defined to formalize an input/output schema of a Web service operation. Three data mediation algorithms are developed to handle the data heterogeneities arising during Web service composition. The data mediation algorithms analyze the schemas of the inputs and outputs of service operations and consider the structures of the schemas. A typed representation for our data mediation approach, which formalizes the data mediation problem as a subtype-checking problem, is presented. An evaluation is performed to study the effectiveness of different data mediation and service suggestion algorithms used to assist designers composing Web services.

Keywords— Web service composition, data mediation, service suggestions, semantic annotations, SAWSDL

I. INTRODUCTION

Web services provide a standard way to publish, discover and invoke diverse software or applications distributed on the Internet. However, in many cases, a complex task may require several Web services working together. Web service composition targets this issue by reusing existing Web services and composing them into a process. Over the last decade, a considerable amount of research on Web service composition has occurred. Part of the work in this area is concerned with making it easier to design Web service compositions by automating portions of the design process.

To compose services that function correctly together, a designer must handle many problems, such as which Web services should be selected, how the services should be connected and how the inputs of each Web service should be fed by the outputs of the preceding Web services.

The focus of this work is on the development of techniques and algorithms that provide enough automation to significantly reduce the human effort required for Web service composition. The approach proposed in this paper is to make timely and effective service suggestions and handle

data mappings during the design process. Three data mediation algorithms are presented to establish data mappings and support the suggestion algorithms. An evaluation has been performed to compare the performance of the three data mediation algorithms.

Using the Semantic Annotation for WSDL and XML Schema (SAWSDL) W3C standard [1], semantics are used by service suggestion algorithms to assist a human designer in composing Web services. The approach proposed in this work can utilize various types of annotations. The first type is annotation on inputs and outputs. These annotations are model references to ontological concepts that capture the notion of a semantic type with a well-defined notion of subsumption determined by description logic reasoning. Another type of annotation addresses the functionality of a Web service operation. Further annotations may be provided to specify lifting and lowering schema mappings that enable more Web services to communicate with one another. Finally, if provided, preconditions and effects can be used for additional checking and/or local planning. Our approach can work with varying levels of semantic specifications that Web service providers and process designers are willing to specify. Additionally, we present a study of the effectiveness of the various types of annotations.

The rest of this paper is organized as follows: Section II describes the Web service model utilized by the algorithms presented in this paper. An overview of the Web service composition approach using service suggestions is presented in section III. Data mediation is explained in detail in section IV. Section V presents the implementation of the system as well as three evaluations along with their outcomes. The related work is discussed in section VI and is followed by conclusions and future work in section VII.

II. SERVICE MODEL

A Web service model should precisely and abstractly represent the information about a Web service that is relevant for composing a Web service process. The model presented here can be considered as a refinement of the notion of semantic templates developed in our prior work on METEOR-S [2]. This model defines a semantic template for a Web service in terms of the following aspects:

interface/porttype, operation, input, output, fault types and QoS specifications. We refine the model by expanding the specifications for input/output to a level of detail sufficient to allow more precise matching of inputs and outputs and to support a form of type safety.

A. Web Service Model

A Web service usually has one or more operations. The input, output, precondition and effects (IOPE) are generally used to describe a Web service operation. Moreover, the syntactic name and the semantic annotation of a Web service operation may indicate its functionality. We formally model a Web service as a set of semantically annotated operations (SOP). Therefore, a Semantic Web Service with n operations is modeled as follows:

$$SWS = \{SOP_j \mid SOP_j : IOFPE, 1 \leq j \leq n\} \quad (1)$$

where $IOFPE = \langle input, output, functionality, precondition, effects \rangle$

B. Graph Model of Input/Output

In this subsection, we propose a graphical model to represent WSDL/XSD data structures for Web service inputs and outputs. In choosing an appropriate graphical representation, the obvious candidates are Trees, Directed Acyclic Graphs (DAGs) and general Directed Graphs. For complete generality, the schema specified using WSDL/XSD for an input or output would require a general directed graph, because recursive structures are permitted in XSD. However, since such recursive structures are not common and using general directed graphs will make some of the problems we are trying to solve NP-Hard, we will limit our work to DAGs. More specifically, we use node-labeled DAGs as defined below:

Definition: An **Input/Output DAG (IODAG)** is a node-labeled DAG $G = \langle N, E, md, Metadata \rangle$ that represents the schema of the input or output of a Web service operation described in a service's WSDL/XSD document(s).

- $N = \{n_1, \dots, n_j, \dots, n_m\}$ is a set of nodes in the DAG. Each node of G corresponds to an element defined in the WSDL/XSD document(s).
- $E \subset N \times N$ is a set of directed edges in the DAG. Each edge $e_{ij} \in E$ is defined as a tuple $e_{ij} = \langle n_i, n_j \rangle$ indicating a directed edge from n_i to n_j , where $n_i, n_j \in N$. The edges indicate the relationships between the elements defined in the WSDL/XSD document. For example, the `<message>` element has one or more `<part>` elements in WSDL, so there will be corresponding edges from the `<message>` node to every `<part>` node in the DAG.
- $md: N \rightarrow Metadata$ is a function assigning labels to nodes.
- $Metadata = Name \times Annotation \times XSDtype$

Definition: The **root node** $n_{root} \in N$ of an IODAG has no incoming edge. An IODAG has exactly one root node,

which corresponds to the `<message>` node in a WSDL 1.1 document.

Definition: A **leaf node** $n_{leaf} \in N$ is any node that has no outgoing edge. $N_{leaf} \subseteq N$ denotes the set of leaf nodes. We consider the leaf nodes as the bottom level nodes of an IODAG.

Definition: A **path** in an IODAG is an ordered list of nodes from a leaf node to the root node along the edges in the IODAG.

$$path = (n_1, \dots, n_j, \dots, n_m) \text{ where } n_j \in N, n_1 \in N_{leaf}, \text{ for } j > 1 \ n_j \in N_{nonleaf} \text{ and } n_m = n_{root} \quad (2)$$

For type checking and similarity scoring, we wish to compare the types associated with nodes, rather than the nodes themselves. Depending on whether n_j is a leaf or nonleaf node, we define the corresponding node type T_j as follows:

$$T_j = \begin{cases} md(n_j) \in T_{leaf} & \text{if } n_j \in N_{leaf} \\ \pi_{name, annotation}(md(n_j)) \in T_{nonleaf} & \text{otherwise} \end{cases} \quad (3)$$

Since we have not yet defined a type hierarchy for all *XSDtypes* including complex types, currently, we must project it out of the *md* function.

Definition: A **path type** in an IODAG is an ordered list of types from a leaf node to the root node along the edges in the IODAG.

$$p = (T_1, \dots, T_j, \dots, T_m) \text{ where } T_1 \in T_{leaf} \text{ and for } j > 1 \quad (4) \\ T_j \in T_{nonleaf}$$

III. OUR SERVICE SUGGESTION APPROACH

This work is a continuation of our previous work [3], which focused on service suggestion algorithms. In this section, we present a semi-automatic approach that suggests Web services to aid a user in designing a Web process. When designing a Web service process using a graphical design tool, a user may ask the system to suggest services that can be plugged into the Web process at a particular position and receive a ranked list of candidate services. Conversely, the user may ask for feedback on a service that they have placed in the process.

The service suggestions are computed based on any combination of semantic annotations provided for a Web service, including each operation's input, output, functionality, precondition and effects (IOFPE), as well as relevant syntactic information. Our service suggestion algorithms rank the available Web services and suggest to the users either the top-k services or those services scoring over a certain threshold. Users can ask the system to suggest which service to connect after, before or in the middle of the process being designed. Respectively, these are referred to as forward, backward and bi-directional suggestions.

The ranking score is calculated as a weighted sum of three sub-scores: the data mediation score (S_{dm}), the

functionality score (S_{fn}) and the precondition/effects score (S_{pe}). The calculation of S_{dm} will be presented in section IV.

The functionality score (S_{fn}) indicates the similarity between the user's desired functionality (a selected concept from the ontology) and the functionality annotation of the candidate operation. The similarity measure used is defined in our previous work [3]. In addition, if a user wishes to give keywords for the functionality, these can be matched to the syntactic name of the operation. Therefore, $S_{fn} = w_1 \cdot S_{sem} + w_2 \cdot S_{syn}$.

To calculate the S_{pe} score, the SWI-Prolog engine is used for logical reasoning. (We were planning to use the Rule Interchange Framework (RIF), but due to the lack of supporting tools we are currently using Prolog.) In the case of a forward suggestion, the S_{pe} score is determined by whether the precondition of the candidate operation $SOP_x.pre$ is satisfied by the current state (st).

$$S_{pe} = \begin{cases} 1 & \text{if } SOP_x.pre(st) = true \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The current state (st) is maintained for the current process in, for example, a Prolog knowledge base. The knowledge base can be queried to check preconditions. After a candidate operation is selected for addition to the process, the current state (st) will be updated to a new state (st'). The new state (st') is determined by applying the candidate operation's (SOP_x) effects to the current state (st).

IV. DATA MEDIATION

Data mediation is one of the major challenges for both semi-automatic and automatic Web service composition. Correctly feeding the outputs of one or more Web services into the inputs of another Web service without human intervention is still an open problem. Furthermore, deciding which output to map to a specific input is not easy for users. Therefore, we have developed a data mediation approach that tries to find automatically the optimal mappings between outputs and inputs.

A. Leaf-based Data Mediation

The simplest form of communication between two service operations happens when one operation sends its output values only to the succeeding operation, which gets its input only from these values. Our leaf-based approach tackles this simply by directly looking for a matching element from the output for each element of the input. It works in two stages:

In the first stage, we parse the WSDL/XSD documents to create IODAG graphical representations: one DAG for the input and one DAG for the output. (In the more general case of having multiple preceding services, the output would be represented as a list of DAGs).

The second stage of our leaf-based approach involves a form of graph matching. Each input must be paired to some output in a way to maximize the overall quality of the match.

The approach naively ignores the nonleaf nodes and just collects all of the leaf nodes to form two sets: one for the output and one for the input. Moreover, it finds the best matching output element for each input element (it is fine if some of the output elements are not used). If we impose the restriction that all the matches must be one to one (exclusive), this problem becomes a weighted bipartite graph problem, where the edge weights are the matching scores between every pair of elements. Thus, it can be solved by a typical weighted bipartite graph algorithm such as the weighted Hungarian (also known as the Kuhn-Munkres (KM)) algorithm [4]. Otherwise, in the shared case, it can be solved by a simpler matching algorithm that runs in $O(mn)$ time, where m is the number of leaves in the output and n is the number of leaves in the input (note, this ignores the time taken in computing the similarity measures). This is more efficient than the KM algorithm, which runs in cubic time [4]. The shared and exclusive formulations are defined in formulas (6) and (7), respectively, where x_{ij} indicates whether the output-to-input pairing $\langle i, j \rangle$ was chosen (1) or not (0).

$$\text{Shared: } \max \left\{ \sum_{i=0}^m \sum_{j=0}^n S_{ij} \cdot x_{ij} \mid \sum_{i=0}^m x_{ij} = 1 \right\} \quad (6)$$

$$\begin{aligned} \text{Exclusive: } \max \left\{ \sum_{i=0}^m \sum_{j=0}^n S_{ij} \cdot x_{ij} \mid \sum_{i=0}^m x_{ij} = 1 \right. \\ \left. \text{and } \sum_{j=0}^n x_{ij} \leq 1 \right\} \quad (7) \end{aligned}$$

where

$$S_{ij} = S(T_i, T_j) = w_1 \cdot \text{conSim}(T_i.OWLtype, T_j.OWLtype) + w_2 \cdot \text{synSim}(T_i.name, T_j.name) \quad (8)$$

is the similarity score between types T_i and T_j , where $w_1 = w_2 = 0.5$ are the weights and conSim and synSim are the concept similarity and syntactic similarity, respectively. For a detailed explanation of the components of the similarity measures, see our previous work [3].

The leaf-based data mediation score (S_{dm}) is the weighted average of the type similarity scores S_{ij} of the n matched pairs of nodes in the optimal matching. The weighted average is computed using formula (14), see section IV.C.

B. Structure-based Data Mediation

The leaf-based data mediation approach considers only the leaf nodes in the IODAG, which leads to the following questions: Are all the other nonleaf nodes useless for data mediation? Does the structure of the IODAGs not matter for data mediation? To address these questions, we consider another approach for data mediation, namely structure-based data mediation, which utilizes a sub-graph homeomorphism algorithm. In our structure-based data mediation, determination of the similarity between the output of a preceding operation and the input of a succeeding operation is modeled as a sub-graph homeomorphism problem between two DAGs: out.IODAG for output and in.IODAG for input. In other words, out.IODAG is sub-graph

homeomorphic to in.IODAG, if out.IODAG contains a sub-graph that is homeomorphic to in.IODAG. The homeomorphism mapping from out.IODAG to in.IODAG specifies how the input of the succeeding operation will be fed by the outputs of the preceding operation.

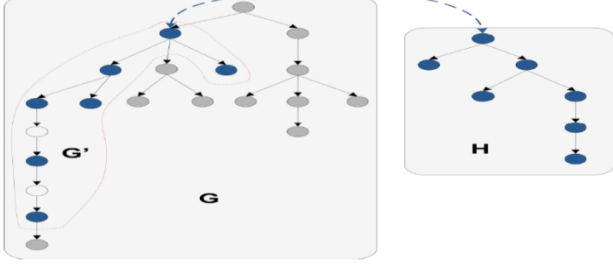


Figure 1. Graph G is sub-graph homeomorphic to graph H

In graph theory, a graph homeomorphism [5] preserves topological properties between two graphs. Specifically, a homeomorphism mapping between two graphs preserves the ancestor relations between the nodes in a graph.

Definition: Graph G is **homeomorphic** to graph H if there is an isomorphism from a subdivision of G to a subdivision of H . In other words, if there exists a graph G' that is obtained by performing a sequence of subdividing and smoothing operations on graph G , and the graphs G' and H are isomorphic graphs, then the graphs G and H are homeomorphic graphs.

Definition: Two graphs, $G = (N, E)$ and $G' = (N', E')$, are **isomorphic** if there exists a bijection $f: N \rightarrow N'$ such that $(n_i, n_j) \in E$ if and only if $(f(n_i), f(n_j)) \in E'$.

Definition: Letting $e = \langle n_i, n_j \rangle$ be an edge of G , the **subdividing** operation adds a new node n_k to G and replaces edge e with two new edges $\langle n_i, n_k \rangle$ and $\langle n_k, n_j \rangle$.

$$\text{subdividing } (G, \langle n_i, n_j \rangle) = G(N \cup \{n_k\}, E - \{\langle n_i, n_j \rangle\} \cup \{\langle n_i, n_k \rangle, \langle n_k, n_j \rangle\}) \quad (9)$$

Definition: Letting n_k be a node of degree of two, such that two edges $e' = \langle n_i, n_k \rangle$ and $e'' = \langle n_k, n_j \rangle$ meet at n_k , the **smoothing** operation replaces edges e' and e'' with a new edge $e = \langle n_i, n_j \rangle$.

$$\text{smoothing } (G, n_k) = G(N - \{n_k\}, E \cup \{\langle n_i, n_j \rangle\} - \{\langle n_i, n_k \rangle, \langle n_k, n_j \rangle\}) \quad (10)$$

Definition: Graph G is **sub-graph homeomorphic** to graph H , if G contains a sub-graph G' that is homeomorphic to H .

The decision problem for sub-graph homeomorphism is known to be NP-complete [6]. However, when the problem is restricted to be a sub-tree homeomorphism (which is often the case for an IODAG representing input or output of a Web service operation), the algorithm to solve it can be efficient, e.g., a cubic time algorithm is presented in [7]. Therefore, we will begin with the case in which the IODAGs are trees and apply the approximate labeled sub-tree homeomorphism

(ALSH) algorithm from [7] to our data mediation problem. The ALSH algorithm takes two trees G and H as parameters and returns the root of the sub-tree of G that has the highest similarity score to H . This score will be taken as the structure-based data mediation score (S_{dm}).

C. Path-based Data Mediation

The leaf-based approach is simple and efficient, but ignores the structure of the input and output, whereas the structure-based approach considers the entire structure, but is more complex and may rule out some useful suggestions. Our third approach decomposes the input and output into many simple paths, rather than the sub-trees used in the structure-based approach.

While the leaves of a DAG provide important information, certainly the full path from a leaf to the root of the DAG should provide more useful information. Comparing full paths is somewhat like comparing sub-structures of the overall DAG. In the worst-case, the number of paths in a DAG may be exponentially large in terms of the number of nodes $|N|$. However, when the structure of the IODAG is restricted to be planar as is often the case with common data structures, the number of paths is bounded by a polynomial [8]. This allows the development of polynomial-time algorithms for path matching.

In our path-based data mediation approach, the input and output of a Web service operation will be represented as semantic ‘structural’ data types based on the IODAG. (Note, structural type equivalence, e.g., as in Modula-3 and OCaml, is more general and complex than conventional name equivalence found in most programming languages). If the output type of an operation is a *subtype* of the input type of another operation, then the output can be fed to the input in a type safe manner. Our type representation scheme contains the definition of a path-set type and four rules to judge the subtype relationships between two types.

Definition: A **path-set type** is the set of all path types in an IODAG.

$$T_{PS} = \{p_j \mid j \in \{1, \dots, n\}\} \quad (11)$$

where n is the total number of paths in the IODAG.

The four subtyping rules used for analyzing IODAGs are defined below:

1. *Rule for leaf types T and T' :* T' is a subtype of T if $T'.annotation$ is subsumed by $T.annotation$ and $T'.XSDtype$ is a subtype of $T.XSDtype$.
2. *Rule for nonleaf types T and T' :* T' is a subtype of T if $T'.annotation$ is subsumed by $T.annotation$.
3. *Rule for path types p and p' :* p' is a subtype of p if starting from the first pair of elements in p and p' , each element in p' is a subtype of the corresponding element in p .
4. *Rule for path-set types T_{PS} and T'_{PS} :* T'_{PS} is a subtype of T_{PS} if for every element p in T_{PS} , there exists an element p' in T'_{PS} such that p' is a subtype of p . (Note, in future work we plan to consider the case where an output path may not be shared.)

The last rule is used as a form of structural level type checking within our path-based data mediation approach. The existence of a subtype relationship between the output type $T_{PS'}$ and the input type T_{PS} indicates that it is safe to directly pass the output to the input. The path-set type decomposes the input and output into many paths based on the IODAGs. If for each path of type p of the input IODAG of an operation, we can find a matching path of type p' among all the output IODAGs, all of the input values of the correct types are available for invoking the operation.

Beyond just type compatibility checking, the path-based data mediation score S_{dm} is also calculated. The score between two paths (SP) is the weighted sum of the scores between each pair of nodes of the two paths as shown in formula (12)

$$SP(p, p') = \sum_{j=1}^L w_j \cdot S(T_j, T'_j) \quad (12)$$

where L is the length of the shorter of the two paths and T_j and T'_j are the j^{th} node types (starting from the leaf) in the path types p and p' , respectively. A geometric sequence is used for all these weights, decreasing from the leaf node to the root node ($w_1 + \dots + w_L = 1$).

Given a path type p in an input IODAG, we find the best matching path type p' in an output IODAG. The score for this best match BSP is defined in formula (13).

$$BSP(p) = \max \{SP(p, p') \mid p' \text{ is from out.IODAG}\} \quad (13)$$

S_{dm} is the weighted sum of the best path scores BSP for the paths in the optimally matched input, where the matching is between the output and the input as shown in formula (14).

$$S_{dm} = \sum_{i=1}^3 \left[w_i \cdot \sum_{j=1}^{n_i} BSP(p_{ij}) \right] \quad (14)$$

where p_{ij} is the path type for the i^{th} input that is either 1 (required), 2 (unknown) or 3 (optional) and n_1 , n_2 and n_3 are the number of required, unknown and optional inputs, respectively ($n_1 + n_2 + n_3 = n$). Inputs to a Web service operation may be specified as required or optional in the WSDL/XSD documents (unknown indicates our software could not determine whether the input was required or optional). Since required inputs tend to be more significant than optional ones, the weights are determined as follows: $w_1 = 1/(n_1 + .8n_2 + .2n_3)$, $w_2 = .8/(n_1 + .8n_2 + .2n_3)$ and $w_3 = .2/(n_1 + .8n_2 + .2n_3)$.

V. IMPLEMENTATION AND EVALUATION

A. Implementation

Our system supporting semi-automatic composition has two major functional components: a service suggestion engine and a data mediation engine as shown in Fig. 2. In addition, several utility components exist to support the service and data mediation engines, i.e., parsers, similarity measures and a knowledge base and its management module. The service suggestion engine invokes the data mediation

engine as a part of its ranking procedure. The data mediation engine can work independently with or without the service suggestion engine. The Java-based implementation of the two engines is currently being coupled with the Galaxy workflow system that is used for designing bioinformatics workflows [9]. Suggestions are provided whenever the designer wishes to have assistance. The system will respond with a top-k list of operations ranked by a score between 0 and 1 as well as an indication of type safety.

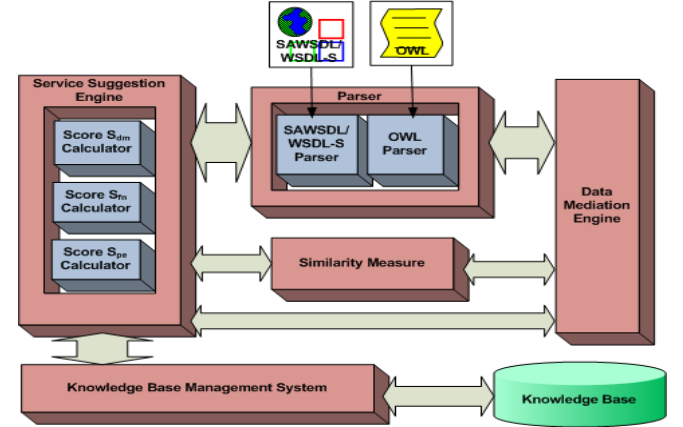


Figure 2. System Architecture

B. Evaluation

The objectives of our evaluation are three-fold: 1) a performance comparison of the three data mediation algorithms to assess the impact of the input/output structure on performance, 2) a study of the effectiveness of various types of semantic annotations and 3) a performance comparison of the forward and bidirectional suggestion algorithms.

1) *Evaluation setup*: Fifty Web service operations are involved in this evaluation. They are from six Web services including the WU-BLAST and ClustalW2 Web services from the European Bioinformatics Institute (EBI)¹. Two ontologies, EMBRACE Data and Methods (EDAM)² and Ontology for Biomedical Investigations (OBI)³, are used/enriched for annotating the Web services. All of the SAWSDL/WSDL-S and OWL files used in this evaluation can be downloaded from the following Website <http://mango.ctegd.uga.edu/jkissingLab/SWS/Wsannotation/wsdls.html>. A bioinformatics process that includes seven Web service operations is used as a motivating scenario (see Fig. 3). The process aims at finding sequences related to a given protein sequence using the WU-BLAST Web service. Sequence results are obtained and then used as input for a multiple sequence alignment using the ClustalW2 Web service.

¹ EBI Web services: <http://www.ebi.ac.uk/Tools/webservices/>

² EMBRACE Data and Methods (EDAM): <http://sourceforge.net/projects/edamontology/>

³ Ontology for Biomedical Investigations (OBI): <http://purl.obolibrary.org/obo/obi>

2) *Comparison of three data mediation algorithms*: The process depicted in Fig. 3 includes seven Web service operations. Suggestions are requested for every operation of the process except the first, which must be chosen by the designer. For each request, the fifty candidate operations have been ranked empirically by three expert human evaluators. Since designing a Web service process is complex, it is difficult to determine precisely if a service is correct. Therefore, we simply compare how well our suggestion algorithms compare to the human evaluators in terms of the degree of overlap of the top-k suggestions. We do this by comparing each of the algorithms with the consensus that averages the three human rankings. Fig. 4 shows the degree of overlap ($|\text{top-k}_{\text{algorithm}} \cap \text{top-k}_{\text{consensus}}|/k$) for the three data mediation algorithms: the leaf-based, structure-based and path-based algorithms. The degrees of overlap are calculated for the top-5, top-10, top-15 and top-20 suggestions as well as for the three data mediation algorithms, as well as for the three human rankings and a random ranking. The results indicate that the path-based data mediation algorithm has a higher degree of overlap than the other two algorithms, which demonstrates that the structure of input/output does affect the performance. The structure-based data mediation algorithm is more complicated and may rule out some useful compositions, thereby resulting in lower performance, although it is still substantially better than the leaf-based data mediation algorithm.

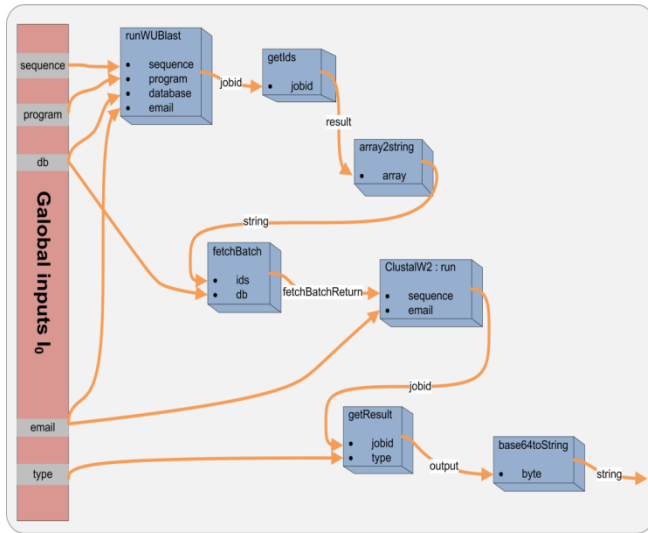


Figure 3. Web process for evaluation scenario

3) *Comparison of semantic annotations*: Complicated semantic annotations might discourage some users from utilizing the semantics-based approaches we investigate. Since our algorithms can work with various types and levels of semantic annotations of Web services, the goal of this evaluation is to study the effectiveness of different types of semantic annotations. Table 1 lists 12 different cases of

semantic annotations evaluated in this experiment (case 0 is effectively random). The syntax and semantic annotations of input/output are used for calculating S_{dm} . The semantic annotations for functionalities are used for calculating S_{fn} . Finally, the precondition and effects are the annotations used for calculating S_{pe} .

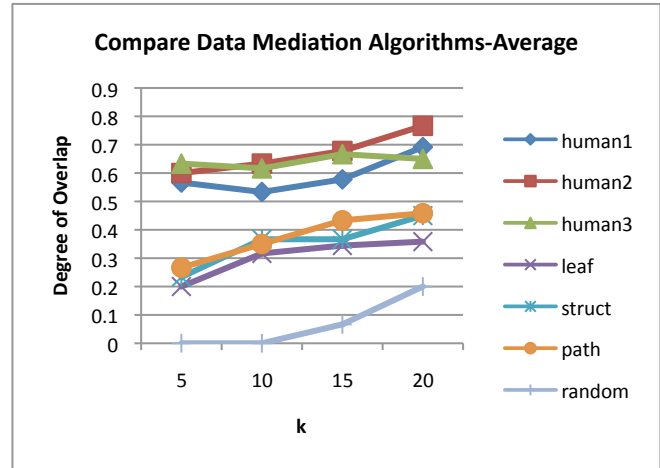


Figure 4. Comparison of Data Mediation Algorithms

The results of this experiment are shown in Fig. 5. Overall, complicated annotations result in higher degrees of overlap, e.g., averaging the top-5 and top-10 the best cases in order are 11, 8, 9, 10 and 5. However, of these, the precondition/effects that require writing formal descriptions in a logic language can be difficult to specify. Therefore, some users may want to avoid such complicated annotations, but still keep acceptable performance. In this experiment, cases 2, 3 and 6 represent the pure annotations on input/output, functionality and precondition/effects, respectively.

Table 1. Different cases of semantic annotations

Information	Syntax used for S_{dm}	Semantic annotations for S_{dm}	Semantic annotations for S_{fn}	Semantic annotations for S_{pe}
Case 0	No	No	No	No
Case 1	Yes	No	No	No
Case 2	Yes	Yes	No	No
Case 3	No	No	Yes	No
Case 4	Yes	No	Yes	No
Case 5	Yes	Yes	Yes	No
Case 6	No	No	No	Yes
Case 7	Yes	No	No	Yes
Case 8	Yes	Yes	No	Yes
Case 9	No	No	Yes	Yes
Case 10	Yes	No	Yes	Yes
Case 11	Yes	Yes	Yes	Yes

The results in Fig. 5 indicate that the relative contributions towards suggesting correct Web services are ordered as follows: functionality annotations (case 3) contribute the most, followed by input/output annotations (case 2) and finally precondition/effects annotations (case 6). Even better is to try combinations such as cases 5 (ranked 5th) or 4 (ranked 6th). Note, neither of these two cases requires

writing complex annotations for precondition/effects, i.e., they require less effort, but still may have sufficient performance and hence may be good choices.

4) *Comparison of forward and bidirectional suggestion algorithms*: Our service suggestion algorithms can make forward, backward and bidirectional suggestions. Since forward and backward suggestions are similar except for the difference in direction, we decided to just compare the performance of forward and bidirectional suggestion algorithms. Fig. 6 depicts the degree of overlap of the top-5, top-10, top-15 and top-20 suggestions made by the forward and bidirectional (2-way) suggestion algorithms, as well as the suggestions made by the three human evaluators and a random ranking. In general, the results of this evaluation show that the forward and bidirectional suggestion algorithms have roughly similar performance, and that further experimentation is needed to differentiate them.

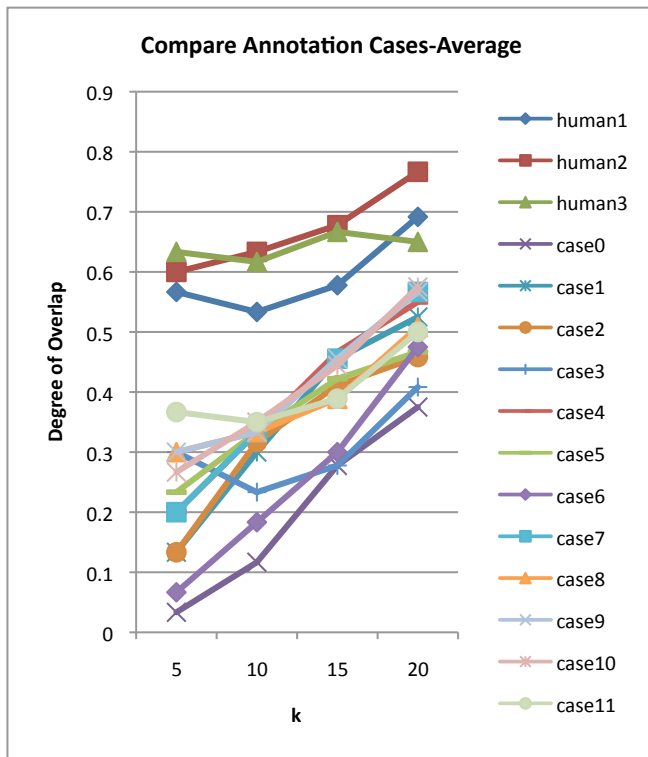


Figure 5. Compare Different Annotation Case

VI. RELATED WORK

This paper proposes a semi-automatic approach for Web service composition, including both data mediation and service suggestion algorithms. Some other approaches also propose to build compositions semi-automatically by making service suggestions/selections: The approach described by Xu et al. [10] helps users select appropriate services by analyzing only dynamic semantic association between services. Kim et al. [11] developed a novel tool for finding errors or deficiencies in Web process designs and providing written suggestions for fixing the problems. Their approach

focuses on finding deficiencies and does not consider data mediation issues in detail. The approach presented in [12] selects services based on QoS parameters, so their selection algorithm is based on non-functional requirements. Michael et al. [13] rank and suggest suitable Web services to the user by using a lazy breadth-first search over an implicit graph of the service space based on the information presented in the Moby service ontology. Unlike our approach their suggestion algorithm does not consider the parameters that are used in our data mediation, e.g., structure of the message schema, XSD type compatibility, preconditions and effects. None of these approaches provide any mechanism to tackle data mediation problems.

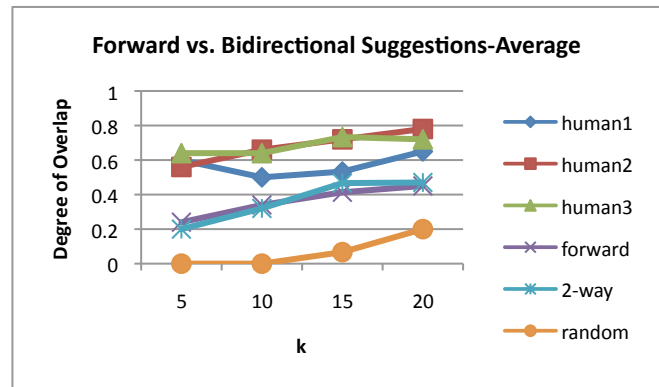


Figure 6. Forward vs. Bidirectional Suggestion Algorithms

Our data mediation approach performs semantic structural type checking to handle the heterogeneities between inputs and outputs of operations. Some other semi-automatic approaches also handle the data mediation problems: [14] uses OWL-S to describe Web services, more specifically, the OWL-S profile sub-ontology, and the data mediation problems are handled by ontology mapping between the sub-ontologies. However, ontology mapping is still an open problem for the Semantic Web. Gao et al. [15] proposed a data mediation approach that is based on matching the schemas of input and output messages, but it only compares the syntax of the message schema without semantics, so it is not able to resolve the semantic heterogeneities between outputs and inputs of the operations. Lebreton et al. [16] describe an OWL-S based approach, which focuses on verifying a given Web process based on data mediation. Its data mediation algorithm only considers identical, generalized and specialized matching at the elemental level, which correspond to our exact, subsumed and subsumed-by matching implemented in our similarity measures [3]. We further compare the properties of the semantic concepts with their ranges, cardinalities, etc. Moreover, the data mediation approach in [16] ignores the structure of the schema of the inputs and outputs and does not perform any type checking.

VII. CONCLUSIONS AND FUTURE WORK

A. Conclusions

This work seeks to aid users trying to compose Web services into a process by providing service suggestions. The

suggestion scores are calculated based on important aspects of Web services, i.e., input/output specifications, functionality and precondition/effects (IOFPE). The service suggestion algorithms can function with/without various types of semantic annotations. An evaluation has been performed to compare the effectiveness of different combinations of annotations, which shows that complete annotations result in higher degrees of overlap. However, considering the effort required to generate fully complete semantic annotations, some less complex cases with acceptable results may be a better choice, e.g., semantic annotations on input/output and functionality.

Three data mediation algorithms, leaf-based, structure-based and path-based are developed to address data heterogeneities in process design, and provide support for service suggestions. Type theory has been studied and the data mediation problem is formalized as a semantic structural subtype compatibility problem. The input/output of a Web service operation is formalized as a structural type, which combines the structural, semantic and syntactic information together. Type checking is performed at both the elemental and structural levels.

The path-based and structure-based data mediation algorithms take into consideration the structure of a Web service operation's inputs and outputs. The path-based algorithm uses all of the paths in an IODAG for making comparisons matching the inputs and outputs. The structure-based algorithm uses a variant of graph isomorphism called sub-tree homeomorphism, which attempts to map the input onto the output. In addition, a simple non-structural data mediation algorithm that just compares leaves is provided. An evaluation is presented to compare these three data mediation algorithms. The results indicate that the path-based data mediation algorithm performs the best, followed by the structure-based and finally the leaf-based algorithm.

B. Future Work

The current service suggestion algorithms consider only functional requirements, such as functionality, precondition, effects, input/output, etc. In the future, some non-functional requirements can be taken into account for service suggestions, e.g., QoS parameters, user's preference, etc. For the path-based data mediation algorithm, a path alignment algorithm or a path-matching algorithm allowing gaps may provide better matching results. Another improvement can be made by annotating Web services with properties defined in an ontology. Currently, all the concepts used to annotate Web services are classes in an ontology. Annotation of properties requires further extension of the similarity measures, such as measuring the similarity between a class and a property defined in an ontology. Finally, in addition to the sub-tree homeomorphism algorithm, we also plan to evaluate a sub-tree homomorphism algorithm that differs from this work in that it provides no subdividing or smoothing operations. We also plan to compare our work more fully with other work that includes type graphs [17], which apparently do not consider issues such as subsumption or subtyping in their matching algorithms.

Acknowledgements: Funding provided by NIH R01 GM093132.

REFERENCES

- [1] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic annotations for wsdl and xml schema," *IEEE Internet Computing*, pp. 60-67, 2007.
- [2] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint driven Web service composition in METEOR-S," in *Proceedings. 2004 IEEE International Conference on Services Computing, 2004. (SCC 2004)*, Shanghai, China, pp. 23 - 30, 15-18 Sept. 2004.
- [3] R. Wang, S. Ganjoo, J. Miller, and E. Kraemer, "Ranking-Based Suggestion Algorithms for Semantic Web Service Composition," in *Proceedings of 2010 IEEE International Workshop on Web X.o (WebX) in conjunction with the 2010 IEEE International Conference on Web Services (ICWS)*, Miami, Florida, 2010, pp. 606-613, July 2010.
- [4] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, pp. 32-38, March 1957.
- [5] J. L. Gross and J. Yellen, *Graph Theory and Its Applications, Discrete Mathematics and Its Applications (2nd ed.)*. Boca Raton, FL: Chapman & Hall/CRC, 2006.
- [6] S. Fortune, J. Hopcroft, and J. Wyllie, "The directed subgraph homeomorphism problem," *Theoretical Computer Science*, vol. 10, pp. 111-121, 1980.
- [7] R. Y. Pinter, O. Rokhlenko, D. Tsur, and M. Ziv-Ukelson, "Approximate labelled subtree homeomorphism," *Journal of Discrete Algorithms*, vol. 6, pp. 480-496, 2008.
- [8] C. Bourke, R. Tewari, and N. V. Vinodchandran, "Directed Planar Reachability Is in Unambiguous Log-Space," *ACM Transaction on Computational Theory*, vol. 1, pp. 1-17, 2009.
- [9] R. Wang, D. Brewer, S. Shastri, S. Swayampakula, J. A. Miller, E. T. Kraemer, and J. C. Kissinger, "Adapting the Galaxy Bioinformatics Tool to Support Semantic Web Service Composition," in *Proceedings of the 3-rd IEEE International Workshop on Scientific Workflows Los Angeles, California, 2009*, pp. 283-290.
- [10] M. Xu, J. Chen, Y. Peng, X. Mei, and C. Liu, "A Dynamic Semantic Association-Based Web Service Composition Method," in *Proceedings of the 2006 IEEE/WIC/ACM, IEEE*, Hong Kong, pp. 666-672, Dec 2006.
- [11] J. Kim, M. Spraragen, and Y. Gil, "An intelligent assistant for interactive workflow composition," in *IUI'04: Proceedings of the 9th international conference on Intelligent user interface*, New York, NY, USA, 2004, pp. 125-131.
- [12] X. Fan, C. Jiang, and X. Fang, "An Efficient Approach to Web Service Selection," in *Web Information Systems and Mining*. vol. 5854, W. Liu, X. Luo, F. Wang, and J. Lei, Eds.: Springer Berlin / Heidelberg, 2009, pp. 271-280.
- [13] D. Michael, P. Rachel, and W. Mark, "Semi-automatic web service composition for the life sciences using the BioMoby semantic web framework," *Journal of Biomedical Informatics, Elsevier Science, San Diego, USA*, vol. 41, pp. 837-847, Oct 2008.
- [14] S. Izza, L. Vincent, and P. Burlat, "Exploiting semantic web services in achieving flexible application integration in the microelectronics field," *Computers in industry*, vol. 59, no. 7, pp. 722-740, 2008.
- [15] A. Gao, D. Yang, and S. Tang, "Web Service Composition Based on Message Schema Analysis," in *Advances in Databases: Concepts, Systems and Applications*. vol. 4443, R. Kotagiri, P. Krishna, M. Mohania, and E. Nantajeewarawat, Eds.: Springer Berlin / Heidelberg, 2007, pp. 918-923.
- [16] N. Lebreton, C. Blanchet, D. Claro, J. Chabaliere, A. Burgun, and O. Dameron, "Verification of parameters semantic compatibility for semi-automatic Web service composition: a generic case study," in *International Conference on Information Integration and Web-based Applications & Services (iiWAS)*, Paris, France, 2010, 8-10 November, 2010.
- [17] G. Spanoudakis and A. Zisman, "Discovering Services During Service-Based System Design using UML," *IEEE Transactions in Software Engineering*, vol. 36(3), pp. 371-389, 2010.