PAPER Special Section on Data Engineering

SAWSDL Service Discovery Based on Fine-Grained Data Semantics

Dengping WEI[†], ***Member***, **Ting WANG**[†a)], ***and*** **Ji WANG**[††], ***Nonmembers***

SUMMARY With the aim to improve the effectiveness of SAWSDL service discovery, this paper proposes a novel discovery method for SAWSDL services, which is based on the matchmaking of so-called fine-grained data semantics that is defined via sets of atomic elements with built-in data types. The fine-grained data semantics can be obtained by a transformation algorithm that decomposes parameters at message level into a set of atomic elements, considering the characteristics of SAWSDL service structure and semantic annotations. Then, a matchmaking algorithm is proposed for the matching of fine-grained data semantics, which avoids the complex and expensive structural matching at the message level. The fine-grained data semantics is transparent to the specific data structure of message-level parameters, therefore, it can help to match successfully similar Web services with different data structures of parameters. Moreover, a comprehensive measure is proposed by considering together several important components of SAWSDL service descriptions at the same time. Finally, this method is evaluated on SAWSDL service discovery test collection SAWSDL-TC2 and compared with other SAWSDL matchmakers. The experimental results show that our method can improve the effectiveness of SAWSDL service discovery with low average query response time. The results imply that fine-grained parameters fit to represent the data semantics of SAWSDL services, especially when data structures of parameters are not important for semantics.

key words: *SAWSDL, service discovery, data semantics, fine-grained*

1. Introduction

The number of the Web services on the Internet has been increasing rapidly since the Service-Oriented Computing (SOC) was proposed. SOC aims to provide a set of methods and tools to support the construction of new applications based on Web services. Service discovery as the first task in the Service-Oriented architecture is to find and select the suitable Web services that satisfy the requirements of the application context at design-time or run-time. The success or failure of the applications based on Web services depends a lot on the effectiveness of service discovery, that is, whether the service discovery agent can find suitable services for the application. Service discovery, therefore, has become the key challenge in the context of Service-Oriented Computing and attracted increasing attention [1]–[7].

The methods of service discovery depend on the considered service description languages, since various service description languages provide different specifications to represent the functional and non-functional semantics of Web services. As a W3C recommendation, Web service description language (WSDL) has become the mainstream XML standards for the inter-operation of Web services. It can specify the operations available together with the structure of data received and output during the execution. However, WSDL can not specify the semantic meanings of the data exchanged between the client and the service. Thus the matchmaker based on WSDL document often returns several useless services to users due to the poor expressiveness of semantics. Fortunately, Semantic Web service (SWS) technique provides a mechanism to describe Web services through explicit semantic meanings. Several SWS ontologies have been proposed since Semantic Web was proposed by Tim Berners-Lee et al. in 2001 [8], like OWL-S [9], WSMO [10], etc. Besides, several semantic-enabled specifications for Web services have also been proposed on top of industry-standard WSDL, e.g., WSDL-S [11], SAWSDL [12]. These various representations lead to appearances of different SWS matchmakers, such as OWLS-iMatcher [13], OWLS-MX [14], WSMO-MX [15], etc.

This paper focuses on the discovery of SAWSDL (Semantic Annotation for WSDL and XML Schema) services for two reasons. First, SAWSDL is a W3C recommendation and bound to be a popular SWS in academic community. Second, SAWSDL is a simple extension of WSDL using two kinds of extensibility elements (i.e., modelReference and schema mapping), and it is easy to be adopted by industry community who follows WSDL as the standard of Web service description.

One of the most important differences with other Semantic Web service ontologies/specifications (e.g., OWLS, WSMO) is that SAWSDL supports various data structure definitions for parameters of operations. However, most matchmakers perform the matching using only selected parts of the SAWSDL description. Taking the matchmakers based on the matching of signature as example, they consider only the matching of input/output parameters at message level. In these matchmakers, a bipartite graph matching is used to measure the similarity between two sets of parameters at message level and a pair of parameters are compared as a whole no matter what kinds of data structures they use. However, similar data may be represented by different data structures in XML Schema, and thus their mismatching may be caused.

In order to overcome these limitations, this paper presents a new matching algorithm for SAWSDL services,

Manuscript received June 7, 2010.
Manuscript revised October 3, 2010.

[†]The authors are with School of Computer, National University of Defense Technology, 410073, Changsha, China.
[††]The author is with National Laboratory for Parallel and Distributed Processing, 410073, Changsha, China.
a) E-mail: tingwang@nudt.edu.cn

DOI: 10.1587/transinf.E94.D.525

which is based on the matchmaking of so-called fine-grained data semantics. Fine-grained data semantics represents the semantics of all the basic data units that services receive and output, which is transparent to the specific data structures by which the parameters are represented. A transformation algorithm is proposed to decompose the data structures of message-level parameters into a set of atomic elements. After the transformation, the matchmaking problem of SAWSDL service signature reduces to the matchmaking problem of corresponding fine-grained data semantics. Meanwhile, a matching algorithm is also proposed for the fine-grained data semantics, in which several matching strategies are given for each kind of description respectively. Finally, this method is evaluated on the SAWSDL service discovery test collection SAWSDL-TC2. The evaluation results show that this method is an effective discovery method for SAWSDL services, with lower computational cost than the discovery methods based on the matching of XML Schema structure.

The paper is organized as follows. Section 2 summarizes related work. Section 3 describes formal definitions for SAWSDL services and depicts the characteristics of SAWSDL services. Section 4 presents the fine-grained data semantics of SAWSDL services and describes the algorithm to generate them. Section 5 presents the similarity measure that is used to compare fine-grained data semantics. Section 6 shows the evaluation of our discovery method, including test collections, evaluation measures and experimental results. Finally, concluding remarks together with future work are presented.

2. Related Work

Current methods for Web service discovery can be mainly categorized into three kinds [16] with respect to the different description used in matchmaking: syntactic, semantic and hybrid methods. Syntactic methods mainly exploit various similarity measures to measure the similarity between the user's requirement and the description of Web service, such as service name, description text, structure description, etc [2]. Semantic discovery methods mainly exploit logic reasoning technique to match semantic annotations of Semantic Web services [1]. Our previous work [3] and [4] proposed a method to enhance the semantic description of Semantic Web service by using the semantic constraints extracted from the description text. The enriched semantic description can improve the effectiveness of discovery. Hybrid methods take into account the semantic matchmaking as well as the syntactic matchmaking, and then rank Web services according to the integrated matching results.

The service search engine Woogle [17] combines multiple sources of evidence to determine the similarity between a pair of Web service operations. The key idea is to cluster parameter names in the collection of Web services into semantically meaningful concepts, which are then compared by using TF/IDF measure. However, the matching of the clustering concepts in Woogle belongs to syntactic methods since there is no logic reasoning used. The method proposed in [5] exploits bipartite graph to match the input and output annotations of services respectively without considering the syntactic descriptions. The work in [18] considers only the matching of WSDL structures and ignores the matching of semantic annotations. The hybrid matchmaker SAWSDL-MX2 [6] uses machine learning technique to integrate both the matching results of structure and semantic signature. URBE [2] proposes a matching algorithm which takes both syntactic and semantic descriptions into account. However, it ignores some specific characteristics from SAWSDL specification and simplifies the matching of data types.

The method in this paper is a kind of hybrid methods, which makes use of both the syntactic and semantic descriptions. The most difference between our method and other methods described above lies in that the granularity of our matching is based on fine-grained parameters rather than the message-level parameters. Compared to URBE [2] and SAWSDL-WA [18] which also consider the structural matching of SAWSDL services, our method abstracts the structural matching into the set-matching of the fine-grained parameters, and it works well when the compared parameters have similar data semantics but are not organized in similar data structures. Although structure may indeed represent some kinds of meanings, similar data may not be matched by the structure matching method when different structures are used to organize the similar data.

3. SAWSDL Services

3.1 Definition of SAWSDL Services

There are three extension attributes defined in SAWSDL, i.e., modelReference, liftingSchemaMapping and loweringSchemaMapping. A model reference may be used as an attribute in every element within WSDL and XML schema. However, SAWSDL defines its meaning only for these components, like wsdl:interface, wsdl:operation, wsdl:fault, xs:element, xs:complexType, xs:simpleType and xs:attribute [12]. In general, model references can be used to help to determine if a service meets the requirements of a client, and schema mappings can be used to address post-discovery issues when the Web service is invoked.

We first introduce the formal notation of a SAWSDL service interface, which describes the most important elements of a SAWSDL service and will be used in the matchmaking algorithm.

Definition 1 (Interface). A SAWSDL service interface is represented as a 4-tuple $interface = \langle iName, iDoc, iAnnotation, OP \rangle$, where *iName* is the unique name, *iDoc* is the description text, *iAnnotation* is the semantic annotation that is the value of attribute modelReference of *Interface* component and *OP* is the set of abstract operations defined in this interface. Each operation in *OP* represents a simple interaction between the client and the service.

Definition 2 (Operation). An operation is represented as

a 5-tuple $op = \langle oName, oDoc, oAnnotation, In, Out\rangle$, where *oName* is the unique name, *oDoc* is the description text, *oAnnotation* is the semantic annotation that is the value of attribute modelReference of *Interface Operation* component, $In = \{i_1, i_2, \ldots, i_n\}$ and $Out = \{o_1, o_2, \ldots, o_m\}$ represent respectively the input and output parameters of this operation. These input/output parameters are usually defined by top-level elements in type system in WSDL document.

Definition 3 (Parameter). A parameter is represented as a 4-tuple $p = \langle pName, pDoc, pAnnotation, pType\rangle$, where

- *pName* is the unique name,
- *pDoc* is the description text,
- *pAnnotation* is the semantic annotation,
- $pType = \langle tName, tAnnotation, tType, SE, baseType\rangle$ is the data type defined in the type system ($\langle types\rangle$ container element) in WSDL document, where
  - *tName* is the unique name of the data type,
  - *tAnnotation* is the semantic annotation,
  - $tType \in$ {bt,st,ct} represents the type of the data type, which can be either built-in data type bt, simple type st or complex type ct in XML Schema,
  - $SE = \{se_1, se_2, \ldots, se_k\}$ represents the set of sub-elements of the data type defined in the type system. If the type is not a complex type, there is no sub-element defined in this type, and thus $SE = \emptyset$.
  - *baseType* is the base data type of the data type. If the data type is a built-in XSD data type, then its base type is the corresponding built-in XSD data type; If the data type is a complex type, then its base type is null; If the data type is a simple type, then its base type is the data type of this simple type.

Each operation takes a set of inputs and produces a set of outputs, which are represented in the signature of the operation in a WSDL document. The signature of an operation, however, provides only the syntactic and structural details (like data types, XML Schemas) of the input/output data. The work in [19] defined the data semantics of Web service as the semantics of the input/output data.

Definition 4 (Data Semantics of Operation). The data semantics of an Operation *op* is represented by a 2-tuple $DS(op) = \langle DS(op.In), DS(op.Out)\rangle = \langle \bigcup_{pi\in op.In}\{pi.pAnnotation\}, \bigcup_{po\in op.Out}\{po.pAnnotation\}\rangle$

3.2 Characteristics of SAWSDL Services

As an extension of WSDL, SAWSDL service description combines the structural features of WSDL and semantic description capabilities of Semantic Web service. Specifically, it has the following features.

- Diversity of data structure. If the data type of a parameter is a complex type, then this parameter may contain several sub-elements which can be encapsulated in different structures. Most matchmakers consider the parameters themselves as the basic units of matching. Thus, similar parameters are often mismatched, since they are represented by different definitions of structures. Furthermore, it is difficult to compare two parameters with different types of data types.
- Uncertainty of semantic annotations. In SAWSDL specification, several components in WSDL can be optionally annotated with semantics using the extension attribute modelReference. Different publishers or annotators of Semantic Web service may annotate the same Web service in different ways. For example, some users may annotate only the interface and the operations of WSDL description, while other users may not annotate the interface and the operation. These uncertain annotating styles will make the same Web service have diverse semantic annotations, and thus bring challenges to the matchmaking of services.
- Unspecified annotating styles. There are two principal techniques for annotating complex types with modelReference attribute, i.e., bottom level and top level annotation. In bottom level annotation, all the member elements and attributes of a complex type will be annotated by adding a modelReference attribute. In top level annotation, the complex types themselves are annotated with a modelReference attribute as a whole. Moreover, a complex type can be annotated at both the top and bottom levels, and these annotations are independent of each other.
- Propagation of model reference. The modelReference properties can be propagated from a type definition component to all element declaration components that are defined with that type, i.e., the modelReference property of an element/attribute declaration also contains the values of the modelReference property from the type definition component referenced by this element/attribute declaration. Therefore, the semantic annotations of a parameter in an operation would contain not only the semantic annotations that annotated on the parameter element itself but also the semantic annotations propagated up to it.

The above characteristics of SAWSDL service make the matchmaking of SAWSDL service different from that of WSDL services and other Semantic Web services. Next, this paper will propose a matchmaker based on the so-called fine-grained data semantics considering all the above features.

4. Fine-Grained Data Semantics

4.1 Parameter-Level Data Semantics and Data Structure

From definition 3, one parameter of an operation is usually defined by the type system in WSDL. Figure 1, for example, shows the output parameters of two operations, in which the top level box "output parameters" represents the set of output parameters, while the bottom level box "types" represents the type system defined in the WSDL document. And

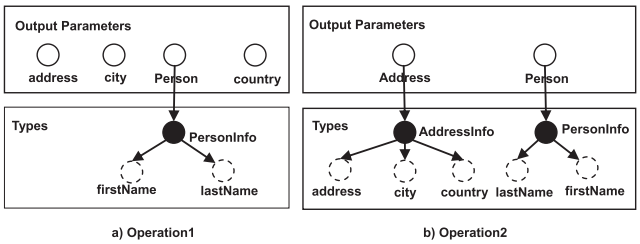

Fig. 1 Examples of output parameters.

the edge between a vertex in "output parameters" and a vertex in "types" shows the data type of parameter. *Operation1* has output parameters {*address*, *city*, *country*, *Person*} and *Operation2* has output parameters {*Address*, *Person*}. Parameter *Person* is a complex type *PersonInfo* and contains sub-elements *FirstName* and *LastName*; Parameter *Address* is a complex type *AddressInfo* and contains sub-elements *address*, *city* and *country*. Obviously, these two operations have very similar semantics, although their output parameters are represented in different data structures due to the flexibility of XML schema definitions.

Most service matchmakers are based on the matching of signatures of operations. In other words, the basic unit considered during matchmaking is the input/output parameters, no matter what kinds of data structures they are. The output matchmaking between *Operation1* and *Operation2* is determined by the matching of two sets of output parameters, i.e., {*address*, *city*, *country*, *Person*} and {*Address*, *Person*}. The matching is usually 1:1 matching, which computes the similarity between each pair of parameters. This kind of matching can not satisfy the requirement of the matching of actual semantics. Taking URBE [2] as an example matchmaker, the similarity between the output parameters shown in Fig. 1 is 0.25 (i.e.,$1/4\times(0+0+0+1)$). The matching value between the parameters *address*, *city* and *country* in *Operation1* and every parameter in *Operation2* is 0, due to the different types of data types used.

However, the good match of the above example is that three parameters *address*, *city* and *country* in *Operation1* are mapped to *Address* in *Operation2*. This is actually a $m : n$ matching. Usually, each parameter can be represented by a XML tree structure and the signature can be represented by a forest. The leaf elements in each tree represent the basic units of data that the operation handles. The complex type, however, is usually used to encapsulate data. Since the structure information of these elements is often transparent to the semantics, we assume in this paper that the semantics of Web service signature can be represented by all the leaf elements in the parameter trees.

Definition 5 (Fine-Grained Parameter).A fine-grained parameter is represented as a 5-tuple $fp = \langle pName, pDoc, pAnnotation, pType, SC\rangle$, where *pName* is the unique name, *pDoc* is the description text, *pAnnotation* is the semantic annotation, and *pType* is the data type of this parameter satisfying $pType.type == bt$ and $SE = \emptyset$. *SC* represents the semantic context of that leaf element, which is represented by a set of semantic annotations.

From the above definition, every leaf element in the parameter tree of an operation is a basic data unit that the operation receives and produces, which is called fine-grained parameter. A fine-grained parameter is a extension of the parameter defined in Definition 3 with the constraint that the data type of the fine-grained parameter is a built-in data type. Meanwhile, if the parent node of the leaf element has a semantic annotation, then the semantic context contains this semantic annotation. If the complex types of SAWSDL service are annotated using Top level annotation, the semantic context can help to enrich the semantics of the leaf elements. Otherwise, these semantic annotation in complex types may be lost after it is decomposed.

In the following, we employ fine-grained parameters to represent the semantics of the signature by transforming each parameter into a set of fine-grained parameters.

Definition 6 (Transformation of Fine-Grained Parameter). A transformation from a parameter $p \in P$ to a set of fine-grained parameters is defined as a function $\tau : P \rightarrow \mathcal{P}(FP)$, where *FP* is the set of fine-grained parameters.

Example 1. In Fig. 1 (a), the set of output parameters at message level is {*address*, *city*, *country*, *Person*}. After transformation, the set of fine-grained output parameters is {*address*, *city*, *country*, *firstName*, *lastName*}, since $\tau(Person) = \{firstName, lastName\}$.

Definition 7 (Fine-Grained Data Semantics). The fingrained data semantics of an Operation *op* is represented by a 2-tuple $FGDS(op) = \langle \bigcup_{pi\in op.In}\{\bigcup_{fpi\in\tau(pi)}\{fpi.pAnnotation\}\}, \bigcup_{po\in op.Out}\{\bigcup_{fpo\in\tau(po)}\{fpo.pAnnotation\}\}\rangle$

In this paper, we consider a more generalized data semantics of operation, which contains not only the explicit data semantics represented by the annotated concepts, but also the implicit semantics represented by the syntactic descriptions of the parameters, like name, description text and data type.

4.2 Fine-Grained Data Generation

By looking into the relationships between data semantics and data structure, it is easy to find that the data semantics of operation is significantly represented by the parameters that the operation receives or produces. The data structure only provides a way to encapsulate data into well-formed data. In this sense, this paper ignores the influence of the data structure and considers only the matching of fine-grained data semantics. To this end, we propose several heuristic rules to decompose the complex-type data structures and transform them into a set of several fine-grained parameters (see Definition 6).

In WSDL 2.0 specification, an *Interface Operation* component describes an abstract operation defined in a given interface. An operation is an interaction with the service, consisting of a set of messages exchanged between the service and the other parties involved in the interaction. And an operation has two required attributes (i.e., name and pat-

tern) and several element children (i.e., input, output, infault, and outfault) that specify the ordinary or fault message types to be used by that operation. The data semantics of the Web service is usually represented by the ordinary message types (i.e., input and output). The number of messages is decided by the used message exchange pattern, and each input/output sub-element represents one message corresponding to the message exchange pattern. The "element" attribute of the input/output element is used to specify the message content that can be either a XML Schema or a non-XML type system. By default, the content model is defined using XML Schema that is defined in the *types* component in WSDL.

Besides the attributes "name" and "modelReference", an element has another important attribute "type" that defines it's data type. A data type [20] can be built-in, simpleType or complexType. The built-in type includes simple data types (e.g., xs:string, xs:decimal, xs:dataTime) as well as derived data types (e.g., xs:integer, xs:short, xs:byte). simpleType is derived from the build-in data type by one of the three means: by restriction, by list and by union. complexType is built by the model group, which can include all the data types (either built-in, simpleType or complexType). There are three kinds of model group elements available: xs:sequence (i.e., the element information items match the particles in sequential order), xs:choice (i.e., the element information items match one of the particles) and xs:all (i.e., the element information items match the particles, in any order).

Definition 6 defines the function of transformation from parameters into fine-grained parameters. The pseudocode of the transformation process is given in Algorithm 1. There are three types of transformations according to the possible data type of the parameter:

- Built-in. If the data type of the transformed parameter is one of the built-in data types (lines 2-9), there is no child element in this parameter. Thus, no transformation is needed for this parameter.
- Simple type. If the data type of the transformed parameter is a simpleType, a simple transformation is needed (lines 10-17). The simpleType element often specifies constraints and information about the values of attributes or the values of text-only elements. The data type of the new fine-grained parameter p' is set as the base type of the original simple data type (line 14). Because of the propagation feature of modelReference attribute, the semantic annotations of the fine-grained parameter p' is set as the logic AND between the semantic annotation of original parameter p and the semantic annotations of the simple type that p refers to(line 15).
- Complex type. In this case, the process of transformation is dependent on the types of group the complex type uses (lines 18-32).
  - sequence (lines 20-23). The sub-elements of the instance of this complex type must appear in sequential order. This order information does not impact the semantics of the set of sub-elements, i.e., the semantics of a complex type definition can be represented by the semantics of all the sub-elements. For example, if two complex type definitions A and B have the same set of child elements but in different orders, the semantics of these two complex types are considered as similar. In other words, the semantics of complex type is transparent to the order of the sub-elements. Therefore, this algorithm will handle each sub-element *sube* defined in the complex type recursively.
  - choice (lines 24-32). The sub-elements of the instance of this complex type can occur zero or one time and the semantics of this complex type is represented by one of the sub-elements. This algorithm takes a hypothesis that the sub-elements defined in one complex type with model group

Algorithm 1 $\tau(p)$: Transformation of Fine-Grained Parameter

Input: a parameter p
Output: a set of fine-grained parameters FP

```
1: FP ← ∅,SC ← ∅
2: if p.pType.type == "bt" then
3:     p′ ← create a new fine-grained parameter
4:     p′.pName ← p.pName
5:     p′.pDoc ← p.pDoc
6:     p′.pType ← p.pType
7:     p′.pAnnotation ← p.pAnnotation
8:     p′.SC ← SC
9:     FP.add(p′)
10: else if p.pType.type == "st" then
11:     p′ ← create a new fine-grained parameter
12:     p′.pName ← p.pName
13:     p′.pDoc ← p.pDoc
14:     p′.pType.tName ← p.pType.baseType
15:     p′.pAnnotation ← p.pAnnotation ⊔ p.pType.tAnnotation
16:     p′.SC ← SC
17:     FP.add(p′)
18: else if p.pType.type == "ct" then
19:     SC.add(p)
20:     if p.pType.constraint == "sequence" then
21:         for each subelement sube ∈ e.eType.SE do
22:             FP.addAll(τ(sube))
23:         end for
24:     else if p.pType.constraint == "choice" then
25:         p′ ← create a new fine-grained parameter
26:         select a subelement sube ∈ p.pType.SE randomly
27:         p′.pAnnotation ← sube.pAnnotation ⊔ p.pType.pAnnotation
28:         p′.pName ← p.pName
29:         p′.pType ← sube.pType
30:         p′.pDoc ← p.pDoc.Concat(sube.pDoc)
31:         p′.SC ← SC
32:         FP.add(p′)
33:     else if p.pType.constraint == "all" then
34:         for each subelement sube ∈ e.eType.SE do
35:             FP.addAll(τ(sube))
36:         end for
37:     end if
38:     SC.delete(p)
39: end if
40: return FP
```

"choice" have very similar semantics in SAWSDL document. Note that if the hypothesis is not true, then different instances of this complex type have different semantics. If an output parameter of a Web service is a complex type t_1 with model group "choice", in which there are two child elements e_1 and e_2 in t_1. This Web service may produce different instances of e_1 or e_2 for clients. The different semantics of e_1 and e_2 make the semantics of Web service uncertain (different semantics of output parameters). That is a paradox. Based on this hypothesis, this algorithm decomposes this kind complex type into one element randomly. The semantic annotation of the new fine-grained parameter is the semantic annotation of the selected sub-element and has the same name as the selected sub-element.

– all (lines 33-35). The "all" element specifies that child elements can appear in any order and that each child element can occur zero or one time. The attribute "minOccurs" specifies that the minimum number of times the element should occur. The default value is 1, although it can be 0 or 1. For simplicity, this algorithm does not distinguish the values of this attribute. The semantics of this kind of complex types is represented by the semantics of all the child elements. Therefore, this algorithm will handle each sub-element *sube* defined in the complex type recursively.

After all the parameters of an operation have been transformed, the signature of the operation can be represented as a set of fine-grained parameters. Next, the matchmaking algorithm will be proposed for the matchmaking of fine-grained parameters.

5. Matchmaking via Fine-Grained Data Semantics

As described in Definition 2 in Sect. 3.1, each operation definition in a service description includes several aspects (such as name, text description, input/output parameters, etc.), which are independent and complementary to each other. To get a comprehensive comparison result, we consider together all these kinds of descriptions for service operation in our matchmaking strategy. And we use and combine several similarity measures to calculate the similarity for operations since different methods fit for comparison of different aspects of service description. For two operations op_r and op_s, the similarity between them is defined as

$$\begin{aligned} Sim(op_r, op_s) = & \\ & \frac{1}{\sum_{i=1}^{5}} \cdot (x_1 \cdot S_{name}(op_r.oName, op_s.oName) \\ & + x_2 \cdot S_{text}(op_r.oDoc, op_s.oDoc)) \\ & + x_3 \cdot S_{annotation}(op_r.oAnnotation, op_s.oAnnotation) \\ & + x_4 \cdot Sim_P(op_r.Out, op_s.Out) \\ & + x_5 \cdot Sim_P(op_s.In, op_r.In)) \end{aligned} \quad (1)$$

where x_i ($i \in \{1, 2, \ldots, 5\}$) is a binary variable (0 or 1) that indicates whether both the request operation op_r and service operation op_s have this part of description information. For example, if both op_r and op_s have description text, then the value of x_2 is 1, otherwise 0. Each similarity measure will be discussed in following sections.

5.1 Similarity of Fine-Grained Parameter

Different kinds of descriptions in Definition 2 have their own specific characteristics and data formats, therefore, different kinds of similarity measures are needed to compute the similarity values. For example, the name is usually a concatenated string and thus string similarity measures are needed for the comparison of operation names, while set similarity measures are needed to compare the set of input/output parameters. Furthermore, there may exist several similarity measures available for the comparison of a certain kind of description, e.g., average string and Jaro coefficient for the comparison of strings. Besides, the effectiveness of different similarity measures is usually dependent on the application context [21]. Therefore, we make use of different similarity measures for the comparison of different aspects of service description.

5.1.1 Name Similarity

The attribute 'name' may appear in every component in SAWSDL definition, such as service name, interface name, operation name, element name. From our previous work [22], we find that *Dice's Coefficient* similarity measure fits for the matching of names and works well on the Web service discovery test collection SAWSDL-TC2 and Jena Geography Dataset 50. Therefore, this paper also employ *Dice's Coefficient* to measure the name similarity of each component. The computational formula of name similarity is defined as

$$S_{name}(x, y) = \frac{2 \times |BS(x) \cap BS(y)|}{|BS(x)| + |BS(y)|}$$

where $BS(x)$ returns the set of character bigrams (sequential pairs) in string x, for example, $BS(hello) = \{he, el, ll, lo\}$.

5.1.2 Text Similarity

Each component in SAWSDL description may have a subelement "document", and the value of the element "document" is the natural language description text of that component. Therefore, it needs to compare the description text. The similarity between text $text_1$ and text $text_2$ is measured by the sets of key words found in the text. After the preprocessing such as tokenization, stemming and stop words filtering, the key words of a description text can be obtained. We use *Dice's Coefficient* to compute the similarity between two set of key words $Key(t_1)$ and $Key(t_2)$. The formula of text similarity is defined as

$$S_{text}(t_1, t_2) = \frac{2 \times |Key(t_1) \cap Key(t_2)|}{|Key(t_1)| + |Key(t_2)|}$$

5.1.3 Semantic Annotation Similarity

In SAWSDL document, several components may have an extension attribute "modelReference" and can be annotated with semantic annotations, such as interface "wsdl:interface", "wsdl:operation", "xs:element". To compare these components, semantic annotations need to be compared according to the semantic subsumption relationships in the ontology definition. There are several similarity measures for measuring the similarity of semantic annotations. This paper selects the most effective measure in OWLS-MX that is *Jeson-Shannon information divergence* similarity [23].

$$S_{annotation}(S, R) = \frac{1}{2\log_2} \Sigma_{i=1}^{n} h(p_{i,R}) + h(p_{i,S}) - h(p_{i,R} + p_{i,S})$$

where $p_{i,R}$ represents the probability of the i^{th} index term appearing in R, and $h(x) = -x \log x$.

5.1.4 Data Type Similarity

The work in [2] presented a similarity function for comparing the similarity between data types. However, it considers only a part of built-in data types in XML Schema. In this paper, we extend it to support more data types. And the similarity between two data types are calculated as the complement of the information loss. For more details we refer the reader to [2].

5.1.5 Semantic Context Similarity

To avoid the loss of semantic annotations of the complex types, Algorithm 1 records the semantic annotations of the complex types in the fine-grained parameters. The similarity between two semantic contexts $sc_1 = \{s_1, s_2, \ldots, s_m\}$ and $sc_2 = \{s'_1, s'_2, \ldots, s'_k\}$ is defined as

$$S_{SC}(sc_1, sc_2) = \max_{1 \le i \le n, 1 \le j \le k} S_{node}(s_i, s'_j)$$

where

$$S_{node}(s_i, s'_j) = \lambda_1 \cdot S_{text}(s_i.nName, s'_j.nName) + \lambda_2 \cdot (S_{annotation}(s_i.nAnnotation, s'_j.nAnnotation))$$

where $\lambda_1 + \lambda_2 = 1$, $\lambda_1, \lambda_2 \in [0, 1]$.

5.1.6 Similarity of Fine-Grained Parameter

The similarity between two fine-grained parameters p_r and p_s is defined as

$$\begin{aligned} &Sim_e(p_r, p_s) \\ &= \tfrac{1}{\Sigma_{j=1}^{5} x_j} \cdot (x_1 \cdot S_{name}(p_r.pName, p_s.pName) \\ &+ x_2 \cdot S_{text}(p_r \cdot pDoc, p_s \cdot pDoc) \\ &+ x_3 \cdot S_{annotation}(p_r \cdot pAnnotation, p_s \cdot pAnnotation) \\ &+ x_4 \cdot S_{type}(p_r \cdot p_r.pType, p_s \cdot pType) \\ &+ x_5 \cdot S_{SC}(p_r \cdot SC, p_s \cdot SC)) \end{aligned}$$

Based on this definition, the similarity between two sets of parameters is defined as

$$Sim_P(X, Y) = \frac{1}{|X|} \sum_{x \in X} \max_{y \in Y} Sim_e(x, y)$$

6. Evaluation

6.1 Experimental Dataset

This evaluation uses one of the most popular SAWSDL service discovery test collection SAWSDL-TC2 [†], which is also used as the test collection in Track 2 of the third Semantic Web Service Selection Contest (S3 Contest) [††]. Web services and requests in SAWSDL-TC2 are transformed semi-automatically from the OWLS test collection OWLS-TC 2.2 [†††] through the tool OWLS2WSD [††††].

SAWSDL-TC2 includes 894 Semantic Web services and 26 requests from 7 domains, which are all described in SAWSDL. Each SAWSDL service or request has only one interface and each interface contains only one operation. There is no description text and semantic annotation for the operation itself in every request, i.e., operation has only name, input parameters and output parameters according to the Definition 2. Each parameter of the operation in the request has name, semantic annotation and type definitions according to the Definition 3. The top level annotation style is employed to annotate the complex types in all the services and requests, i.e., only the complex types themselves are annotated with model references. However, the interface, operation and non top-level elements in "types" definition are not annotated.

The previous work in [6] used a subset of SAWSDL-TC2, called SAWSDL-TC2_WA, for evaluation. In this paper, we also use this subset for comparison with the previous work. This subset contains only 14 requests but all the services in SAWSDL-TC2.

6.2 Evaluation Measures

The performance of Web service discovery is often measured by two popular measures in information retrieval, which are precision and recall. Given a request q with n relevant Web services, precision P is the fraction of retrieved relevant Web services (k) to the retrieved Web services (m).

$$P = \frac{|relevantservices \cap retrievedservices|}{|retrievedservices|} = \frac{k}{m}$$

Recall is the fraction of retrieved relevant Web services (k) to the total relevant services (n).

$$R = \frac{|relevantservices \cap retrievedservices|}{|relevantservices|} = \frac{k}{n}$$

[†] http://projects.semwebcentral.org/projects/sawsdl-tc/
[††] http://www-ags.dfki.uni-sb.de/ klusch/s3/html/2009.html
[†††] http://semwebcentral.org/projects/owls-tc/
[††††] http://projects.semwebcentral.org/projects/owls2wsdl/

Precision and recall are single-value metrics based on the whole list of returned services by the matchmaker. However, the order is also important to the matchmakers that return a ranked list of services. Average Precision AP is the average of precisions computed at each point of the relevant services in the ranked list, which emphasizes ranking relevant services higher.

$$AP = \frac{\Sigma_{i=1}^{m} P@i \cdot rel(s_i)}{\Sigma_{i=1}^{m} rel(s_i)}$$

where m is the cardinality of the set of returned services, $P@i$ is the precision only considering the top i returned services, s_i represents the i^{th} service in the returned services list, function $rel(s_i)$ is a binary function on the relevance (e.g., if s_i is a relevant service, then $rel(s_i) = 1$, otherwise $rel(s_i) = 0$).

Mean Average Precision (MAP) is a metric for the whole requests. It is a mean value of average precision of each request in the query set Q.

$$MAP = \frac{\Sigma_{i=1}^{|Q|} AP_i}{|Q|}$$

In our evaluation, we use macro-averaged precision and recall curves to compare the effectiveness of the compared matchmakers. At each recall level, the average precision over all queries is the macro-average precision. However, different queries have various number of relevant services, and it is, therefore, difficult to compute the average precision at a certain recall level. Ceiling interpolation is used to estimate the precision values that are not observed in the relevant sets for some queries at some levels. The number of standard recall levels R_i from 0 to 1 we used in this evaluation is 20 (i.e., $R_i = \frac{i}{20}$). The macro-averaged precision is defined as follows:

$$mAP_i = \frac{\Sigma_{q \in Q} \max\{P' | R' \geq R_i \wedge (R', P') \in S\}}{|Q|}$$

where S is the set of observed (R,P) points, R_i is a standard recall level.

Besides, query response time (QRT) is also used to measure the efficiency of the matchmakers, which is the average time from the request to the response time. Average query response time (AQRT) is the average response time over all queries in the test collection.

6.3 Experimental Results and Analysis

Our matchmaker is called FGDSM (Fine-Grained Data Semantics based Matchmaker). The benchmark matchmakers consist of the variants of SAWSDL-MX [6], including SAWSDL-WA, SAWSDL-M0, SAWSDL-M0+WA and SAWSDL-MX1. SAWSDL-WA exploits the method proposed in WSDL-Analyzer [18] to recursively compute the structural similarity between request and service, which involves the comparison of element name, data types, attributes of structure, etc. SAWSDL-M0 is a pure semantic matchmaker based on the logic matching of the semantic annotations of parameters. There are five degrees for the logic matching, including *Exact*, *Plug-in*, *Subsumes*, *Subsumed-by* and *Fail*. Services are ranked according to their matching degrees in the following decreasing order: *Exact* > *Plug-in* > *Subsumes* > *Subsumed-by* > *Fail*, and the services that share the same logical matching degree are ranked at random. SAWSDL-M0+WA exploits the similarity value computed by SAWSDL-WA to rank the services in a certain degree which are filtered by the logic matchmaker SAWSDL-M0. SAWSDL-MX1 applies the logical matching filters mentioned in SAWSDL-M0 and ranks services that share the same logical matching degree according to the syntactic similarity of semantic annotations.

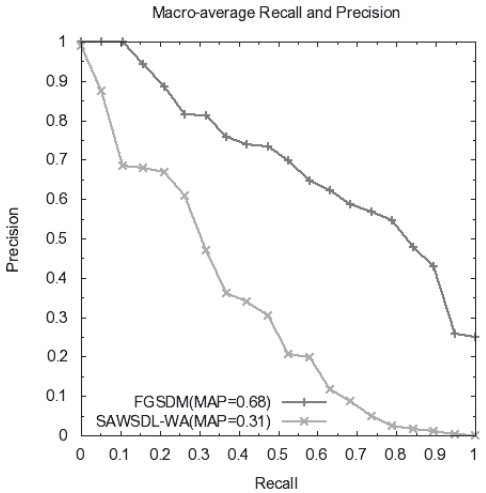

Fig. 2 Performance comparison on SAWSDL-TC2_WA.

Currently, the matchmaker FGDSM has already been deployed in our customizable matchmaker SAWSDL-iMatcher. Meanwhile, a FGDSM plug-in of the general Semantic Web service matchmaker evaluation environment (SME2) is also implemented, and all the experimental results are produced by this plug-in.

Figure 2 shows the macro-averaged precision and recall curves of FGDSM and SAWSDL-WA on the test collection SAWSDL-TC2_WA. The results show that FGDSM outperforms the benchmark matchmaker SAWSDL-WA, that is, FGDSM has higher precision and recall. Since FGDSM performs the matchmaking of the fine-grained data semantics, it works well when the compared parameters have similar data semantics but are not organized in similar data structures, which may not be matched in SAWSDL-WA successfully. Besides, this may indicate that the matchmaker considering both the structure and semantic annotations can improve the effectiveness of matchmaking, compared with the matchmakers that only consider the structure information.

Table 1 summarizes the average precision (AP) and average query response time ($AQRT$) of each benchmark matchmakers on test collection SAWSDL-TC2_WA. The experimental results of SAWSDL-MX1 and SAWSDL-

Table 1 Comparison of AP and ARQT.

	FGDSM	SAWSDL-WA	SAWSDL-MX1	SAWSDL-M0+WA
MAP	0.68	0.31	0.66	0.61
ARQT	2.74s	11.94s	8.17s	15.48s

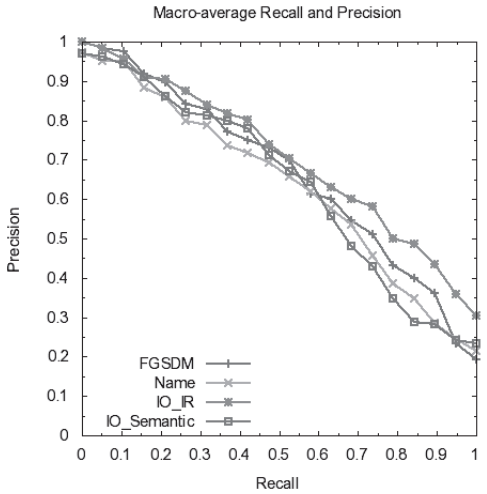

Fig. 3 Performance comparison on SAWSDL-TC2.

M0+WA are referenced from [6]. The results indicate that the structure based matchmaker SAWSDL-WA has lower average precision and higher query response time. FGDSM has average precision of 0.68, which is slightly better than that of SAWSDL-MX1 (MAP = 0.66). Furthermore, since FGDSM transforms the structure information into flatten information set, it has lower query response time by avoiding the comparison of structure.

Figure 3 shows the performance of each matchmakers on the test collection SAWSDL-TC2. "Name" represents the name matching based matchmaker. "IO_IR" compares the semantic annotations of interface and explores *Euclidean Distance* to compare the unfolded concept expressions of semantic annotations. "IO_Semantic" exploits logic matching to measure the similarity between the semantic annotations of interface [22]. The experimental results show that FGDSM outperforms service name based matchmaker and logic matching based matchmaker. When the recall levels are lower than 20%, FGDSM outperforms IO_IR. Otherwise, IO_IR slightly outperforms FGDSM.

The above experimental results indicate that fingrained data semantics based matchmaker can obtain good effectiveness and avoid the high computational complexity. This method can satisfy the user's requirement that the matchmakers should return the services list as good as possible and respond the request as soon as possible.

7. Conclusion and Future Work

This paper presents a SAWSDL service matchmaking method based on fine-grained data semantics. The core of this method is to transform the message-level parameters into fine-grained parameters by considering the characteristics of the SAWSDL structure and annotations. On this basis, we propose a matching measure for the fine-grained parameters, which integrates several parts of SAWSDL service descriptions (including name, description text, semantic annotations, data types and semantic context). Experimental results show that the fine-grained data semantics can help to improve the performance of the matchmaking with low computational cost.

In the future work, we will consider the semantic matchmaking of precondition and effect of Semantic Web services at the fine-grained level.

Acknowledgements

The research is supported by National Grand Fundamental Research Program of China under Grant No.2011CB302603 and the National Natural Science Foundation of China under Grant No.60873097.

References

[1] M. Paolucci, T. Kamasutra, T.R. Payee, and K.P. Sycara, "Semantic matching of web services capabilities," ISWC, LNCS, vol.2342, pp.333–347, Springer, 2002.
[2] P. Plebani and B. Pernici, "Urbe: Web service retrieval based on similarity evaluation," IEEE Trans. Knowl. Data Eng., vol.21, no.11, pp.1629–1642, 2009.
[3] D. Wei, T. Wang, J. Wang, and Y. Chen, "Extracting semantic constraint from description text for semantic web service discovery," ISWC, LNCS, vol.5318, pp.146–161, Springer, 2008.
[4] D. Wei, T. Wang, J. Tang, and J. Wang, "Reducing semantic bias of annotations for semantic web service discovery," J. Southeast University, vol.26, no.1, pp.48–52, 2010.
[5] U. Bellur and R. Kulkarni, "Improved matchmaking algorithm for semantic web services based on bipartite graph matching," IEEE International Conference on Web Services, pp.86–93, IEEE Computer Society, 2007.
[6] M. Klusch, P. Kapahnke, and I. Zinnikus, "Hybrid adaptive web service selection with sawsdl-mx and wsdl-analyzer," ESWC, LNCS, vol.5554, pp.550–564, Springer, 2009.
[7] J. Becker, O. Müller, and M. Woditsch, "An ontology-based natural language service discovery engine - design and experimental evaluation," ECIS, 2010.
[8] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," Scientific American, 2001.
[9] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. Mcllraith, S. Narayanan, M. Paolucci, B. Parsia, P. Terry, E. Sirin, N. Srinivasan, and K. Sycara, "Owl-s: Semantic markup for web services," 2004. http://www.w3.org/Submission/OWL-S/
[10] J.d. Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, E. Oren, A. Polleres, J. Scicluna, and M. Stollberg, "D2v1.3. web service modeling ontology (wsmo)," 2006. http://www.wsmo.org/TR/d2/v1.3/
[11] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.T. Schmidt, A. Sheth, and K. Verma, "Web service semantics - wsdl-s," 2005. http://www.w3.org/Submission/WSDL-S/
[12] J. Farrell and H. Lausen, "Semantic annotations for wsdl and xml schema," 2007. http://www.w3.org/TR/sawsdl/
[13] C. Kiefer and A. Bernstein, "The creation and evaluation of iSPARQL strategies for matchmaking," ESWC, LNCS, vol.5021, pp.463–477, Springer, 2008.
[14] M. Klusch, B. Fries, and K.P. Sycara, "Owls-mx: A hybrid semantic web service matchmaker for owl-s services," J. Web Semantics,

vol.7, no.2, pp.121–133, 2009.
[15] M. Klusch and F. Kaufer, "Wsmo-mx: A hybrid semantic web service matchmaker," Web Intelligence and Agent Systems, vol.7, no.1, pp.23–42, 2009.
[16] M. Klusch, "Semantic service coordination," in CASCOM - Intelligent Service Coordination in the Semantic Web, ed. M. Schumacher, H. Helin, and H. Schuldt, ch. 4, pp.59–104, Springer, 2008.
[17] X. Dong, A.Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," VLDB, pp.372–383, 2004.
[18] I. Zinnikus, H.J. Rupp, and K. Fischer, "Detecting similarities between web service interfaces: The wsdl analyzer," Second International Workshop on Web Services and Interoperability, 2006.
[19] K. Sivashanmugam, A. Sheth, J. Miller, K. Verma, R. Aggarwal, and P. Rajasekaran, "Metadata and semantics for web services and processes," in Datenbanken und Informationssysteme (Databases and Information Systems), pp.245–271, Festschrift zum 60. Geburtstag von Gunter Schlageter, Publication Hagen, 2003.
[20] A.M.P.V. Biron and K. Permanente, "Xml schema part 2: Datatypes second edition (w3c recommendation)," Tech. Rep., W3C, Oct. 2004.
[21] A. Bernstein, E. Kaufmann, C. Buerki, and M. Klein, "How similar is it? towards personalized similarity measures in ontologies," Wirtschaftsinformatik, pp.1347–1366, 2005.
[22] D. Wei, T. Wang, J. Wang, and A. Bernstein, "Sawsdl-imatcher: A cunstomizable and effective semantic web service matchmaker," Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2010 (Under Review).
[23] M. Klusch, B. Fries, and K.P. Sycara, "Automated semantic web service discovery with owls-mx," AAMAS, pp.915–922, ACM, 2006.

Dengping Wei was born in 1981, female, Ph.D. student. The main research interest includes Semantic Web, Web service and information retrieval. School of Computer, National University of Defense Technology, 410073, Changsha, China.

Ting Wang was born in 1970, male, professor. The main research interest includes Semantic Web, information extraction and information retrieval. School of Computer, National University of Defense Technology, 410073, Changsha, China.

Ji Wang was born in 1969, male, professor. The main research interest includes high confidence software technologies, software methodology and software engineering. National Laboratory for Parallel and Distributed Processing, 410073, Changsha, China.