Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(6):2570-2576

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

A web service matching method based on fine-grained data semantics

Yanping Li

Department of Computer and Information Engineering, Heze University, Heze Shandong, China

---

ABSTRACT

The service discovery methods based on structure matching has high computational cost and can not distinguish the Web services with similar semantics but in different structure definitions, due to the flexibility of XML Schema. To solve this problem, a so-called fine-grained data semantics method is proposed, which can be obtained by a transformation algorithm that decomposes parameters at message level into a set of fine-grained data parameters. 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. Experiments show that the proposed method can improve the effectiveness of service discovery.

Keywords: Semantic Web service, Service Discovery, Matchmaking Strategies

---

In recent years, along with the improvement of the related standards of Web service and the gradually mature software platform which support the Web service development, the number of Web services on the Internet is continually growing. Web service discovery is an important part of the Web service structure. Its task is to find the service which can meet customers' demand t from the huge number of server on the Internet, and its core is a service matching algorithm of good performance [1]. The current Web services matching algorithm is mainly divided into grammatical level and semantic level. The core of semantic Web service technology is to use the ontology to give an ontological description of Web service, thus making the semantics much clearer. The semantic Web services technology discovery has a higher accuracy, but the time complexity of ontology reasoning is too high; and, to accurately mark a large number of Web services is a heavy task[2,3]. At present the web services on the internet are growing day by day and the semantic web service technology is still not mature. Therefore, how to further improve the efficiency of the web service discovery becomes an urgent problem to be settled. This paper proposes a decomposition algorithm, which will decomposes various types of data structures that describe WSDL operation information elements into a set of elements with the basic data type, and then turns the SAWSDL service matching based on messages into more fine-grained semantic matching data.

SEMANTIC ANNOTATION FOR WSDL

SAWSDL (Semantic Annotation for WSDL) is the W3C recommended standard improved on the basis of WSDL. Its original version is WSDL-S language proposed by the team Meteor-S in LSDIS laboratory, Georgia University[4,6]. SAWSDL is to extend WSDL elements by using ontology annotation methods to represent the semantic information of Web services. By using external semantic model, service providers can choose their domain ontology to annotate service interface parameters. The SAWSDL documents have both structure characteristics of WSDL documents and semantic description characteristics of Web services, which is shown in detail as follows:

1) structuralized characteristics of messages

In SAWSDL services description, the type of operation messages is by default regarded as the data type of XMLSchema definition. The message type can be the basic type of XML Schema, and also can be the simple type and complex type defined in XML Schema, while the diversity of message types makes the message matches become more complex.

2) uncertainty of semantic annotation
The extended attribute modelReference in SAWSDL specification can make semantic annotation for a variety of elements, and the annotation is optional. This leads to the fact that different semantic Web service publication personnel or the annotation personnel may give the inconsistent annotations to the same semantic web services, in which there exists diversity.

3) ModelReference attribute annotation types about complicated data
There exist two types: from bottom to top and top marking, and the two marking methods are independent of each other and can also exist at the same time. Bottom-up marking method only annotates elements and attributes in definitions of complex types, while the top marking method is only applicable for the complex type itself, not including its elements. This case brings out challenges to the comparison of complex types.

4) the media features of ModelReference attribute annotation
SAWSDL regulates that the ModelReference attribute value of definition data types is spread to the elements which reference these data types. That is to say, the model reference attribute value of element declaration should contain the modelReference attribute values in the data types definition set.

The greatest advantage of SAWSDL is that it is compatible with the WSDL language, and is easy to realize. This is essential for practical application of SWS language, because the existing Web services are all expressed by WSDL language, if we completely abandon WSDL and use the new language to describe Web services, it will spend huge resources. In addition, SAWSDL is also a lightweight language, unrelated to the specific ontology description language, i.e. it can use any ontology language model for semantic annotation, and at the same time support the logic and non - logic ontology model.

FINE-GRAINED DATA PARAMETERS

2.1 Definition of fine-grained data parameter

Definition 1 (parameter types) The parameter types of type systems in a WSDL document can be represented by a five-tuple flow ,that is, eType=〈tName，tAnnotation，tType，SE，baseType〉. Where in, tName is the name of a data type; tAnnotation for semantic annotation of data types; tType ∈{BT, St, CT} represents the type of this data type, namely, BT represents the inside data types, St represents a simple data type, CT stands for complex data types; SE={se1, se2,... Sek} represents the sub-element of defined data types in the type system of the element. If the data type is not a complex one, the set is empty, i.e. SE= Φ; baseType is the basic data type. If such data type is the built-in basic type, its basic type is just itself; if the data is a simple type, the basic type is its defined basic type; if the type is a complex type, its basic type is NULL

The data semantics of web services is represented by its input and output semantics. Thus, the data semantics of Web operation and formalized definition of the fine-grained parameters are given.

Definition 2 (definition of operational data): a data semantics of a OP operation is represented by a two-tuple flow, namely, DS(op)=〈DS(op,In),DS(op,Out)〉=〈∪pi∈op.In{pi.pAnnotation},∪po∈op.Out{po.pAnnotation}〉

Definition 3 (fine-grained parameters) a fine-grained parameter is expressed as a five-tuple flow, i.e. fp=〈pName, pDoc, pAnnotation, pType, SC〉. Wherein, pName is the only name, pDoc is the description text of parameters, pAnnotation is the representation of semantic annotation of parameters. pType is the data type definition of the parameters, which also meet the following relation: pType.type==bt, pType.SE=, and SC represents the semantic context of the parameters, mainly composed by the semantic annotation set.

From the above definition we can see that all leaf elements in every data type definition tree of operation parameters can be called fine-grained parameters, and it is the basic unit of data receiving and generating operation. If the father nodes of leaf elements have a semantic annotation, then according to the propagation characteristics of the modelReference attribute value, the semantic context of this element contains the concept of semantic annotation. If the complex type in SAWSDL services uses the top tagging method, to save the semantic context information is beneficial to rich semantic leaves elements, and reduces loss of information caused by the generation of fine-grained data parameters

Definition 4 (conversion of fine-grained data parameters) The conversion of a parameter P ∈P to the fine-grained parameter set can be defined as a functionτ：P→P(FP), wherein, FP represents the fine-grained parameters set.

Definition 5 (definition of fine-grained data) A data semantics of a OP operation can be represented by a two-tuple

flow, i.e., FGDS(op)= 〈 ∪pi∈op.In{ ∪fpi∈τ(pi) {fpi,pAnnotation}}, ∪po∈op.Out{ ∪fpo∈τ(po) {fpo,pAnnotation}}〉.

2.2 The generation of fine-grained data parameters

The fine-grained data semantics refers to the fact that each operation parameter is used as the data operation unit of the atom, that is, elements which have built-in basic data types. The fine-grained conversion of operation data semantics refers to the decomposition and transformation of the elements which are not the built-in basic data types, thus making the types of every interface parameter be built-in data types XML Scheme, namely the transformation of operation elements which have simple data types and complex data types. Through the transformation, all operation parameters of Web services are the element sets with XML Scheme's built-in basic data types, and each element represents a basic data unit of Web input or output.

With the idea of the above fine-grained data semantic transformation, the element operating at message level e= 〈 eName,eDoc,eAnnotation,eType 〉 is transformed into the fine-grained elements set E′= { e′ | e′= 〈pName,pDoc,pAnnotation,pType,SC 〉, in order to facilitate the matching. Among them, E′is the number of all leaf elements of the real element e. It should be noticed that there appear a new attribute SC in fine-grained elements, which are used to record the context information of the new leaf elements. In order not to lose the semantics of complex data types, this transformation algorithm will preserve semantic path information which generates fine-grained elements as the context information of fine-grained elements, namely, the decomposition path of complex data types.

Definition 6 (the generation path) The generation path of a fine-grained element is the decomposition sequence of complex data types which generate the fine-grained elements: ePath=node1, node2, noden. Moreover, each complex data type is represented by a two-tuple flow, i.e., node=〈nName,nAnnotation〉. Among them, nName is the name of complex data types, and nAnnotation is the semantic annotation of complex data types,

2.3 data semantic decomposition transformation algorithm

The transformation algorithm based on operating elements at the message level: ElementTranslation algorithm as follows:

Algorithm 1：ElementTranslation
（τ(p)）:Transformation of Fine-Grained Parameter

Require: *a parameter p*

Ensure: *a set of fine-grained parameters FP*
1: FP←Φ,SC←Φ,pt←=p
2： ***if*** *p.pType.type==”bt”* ***then***
3: p′.SC←SC
4: FP.add(p′)
5: ***else if*** *p.pType.type==”st”* ***then***
6: p′.pType.tName←p.pType.baseType
7: p′.pAnnotation←p.pAnnotation ∪p.pType.tAnnotation
8: p′.SC←SC
9: FP.add(p′)
10: ***else if*** *p.pType.type == ”ct”* ***then***
11: SC.add(p)
12: ***if*** *p.pType.constraint == “sequence”* ***then***
13: ***for*** *each subelement sube ∈e.eType.SE* ***do***
14: FP.addall（τ(sube)）
15: ***end for***
16: ***else if*** *p.pType.constraint == “choice”* ***then***
17: select a subelement sube ∈p.pType.SE randomly
18: p′.pAnnotation←sube.pAnnotation ∪p.pType.pAnnotation
19: p′.pType←sube.pType
20: p′.pDoc←p.pDoc.Conct(sube.pDoc)
21: p′.SC←SC
22: FP.add(p′)
23: ***else if*** *p.pType.constraint == “all”* ***then***

24: ***for*** *each subelement sube ∈e.eType.SE* ***do***
25: *FP.addAll（τ(sube)）*
26: ***end for***
27: ***end if***
28: *SC.delete(p)*
29: ***end if***
30: ***return*** *FP*

Algorithm 1 converts parameters at message level into the set of fine-grained data. According to possible parameter type, this algorithm deals with three different transformations of data types parameters:

1) built-in basic data type conversion: if the data types of the converted elements is the built-in data types in XML Schema, the message data elements are the basic elements, without conversion, and the path is empty.

2) the simple data types: if the data types of the converted elements are the simple data types, it's need to transform to the elements. The basic types of new elements are set as the basic types of the definition in the original simple data types. The semantic annotation of the new elements is updated as the logic or of both the semantic annotation of original elements and the semantic annotation of its simple reference data types of the original elements. This is because according to the propagation characteristics of modelReference in the SAWSDL specification, the semantic annotation of elements should contain semantic annotation of its reference data type.

3) complex data types: if the data types of the converted elements are complex data types, the algorithm will convert and process the elements of the complex types according to the constraint types. If the constraint type is sequence, it shows that all child nodes of elements of the complex data elements must appear in order. This paper does not consider the semantic difference brought by the order, and proposes that the order has no influence on service discovery task, namely, the semantics of complex data elements can be expressed by all the child nodes. If the constraint type is choice, it shows that the child elements in instances of the complex types appear for 0 or 1 time, that is, the semantics of the complex types will be expressed by semantics of a subset of the elements. Although the SAWSDL specification does not explicitly limit the constraint form of complex data types, the algorithm is based on the assumption, namely, in the usual case, the definitions of the child elements in complex data type choice in the document SAWSDL have similar semantics. If the constraint type is all, it shows that the appearance number of the child elements in the instances of complex types is unknown, either 0 or 1. This constraint type has a "minOccurs" attribute which is used to specify the number of child elements, although the attribute value can be either 0, or 1, the default is 1. For simplicity, the algorithm will not distinguish the effects of this attribute value on the transformation of complex data type elements, that is to say, the semantics of complex data elements can be represented by the semantics of all child elements, therefore, the algorithm will recursively process each child element sube in the complex data types.

OPERATION MATCHING BASED ON FINE-GRAINED PARAMETERS

3.1 Basic attribute matching

1) Name comparison

SAWSDL service description mostly involves the name of each component, such as the service name, the interface name, the operation name, the element name, etc.. In previous work we have found that the similarity measure of Dice 's coefficient[89] is suitable for the service name matching, and has a better effect in the existing Web service data set (SAWSDL-TC2 and Jena Geography Dataset 50). Therefore, this paper adopts Dice 's coeffcient to to measure the similarity of all the component names, and its calculation formula is as follows.

$$S_{name}(x,y) = \frac{2' \left| BS(x) \cap \mathrm{BS}(y) \right|}{\left| BS(x) \right| + \left| \mathrm{BS}(y) \right|} \quad (1)$$

Among them, BS (x) shows the set of all bigrams in a string x, for example, BS (hell0) ={he, El, ll, lo}.

2) Descriptive text comparison

Every component in SAWSDL service description can have child elements"docement", through which the natural language descriptive text of every component can be defined. The similarity of T1 in this paper and T2 can be measured by set by the similarity of the keyword set. Firstly, the root lexical analysis, and stop-word filtration of natural language text will be carried out, and then the Dice 's coeffcient is adopted to calculate the similarity of the two keyword set key (T1) and key (T2).

$$S_{\text{text}}(t_1,t_2)=\frac{2\times\left|key(t_1)\cap key(t_2)\right|}{\left|key(t_1)\right|+\left|key(t_2)\right|} \tag{2}$$

3) Semantic annotation comparison

In the SAWSDL documents, there possibly exists semantic annotation in interface definition (WSDL:interface), operation definition (WSDL:operation), error definition (WSDL:fault), the elements of the type system (xs:element), a complex type (xsLcomplexType), a simple type (xs:simpleType), and attributes (xs:attribute). The similarity measure of these components need to compare the similarity of semantic annotation between these components. Usually these semantic taggings are concepts in the ontology. There are many algorithms to measure the similarity of concepts in the ontology. The paper selected the best similarity metric Jenson-Shannon information divergence[58] in the OWLS-MX match to compare similarity of two semantic annotation:

$$S_{\text{annotation}}(S,R)=\frac{1}{2\log_2}\sum_{i=1}^{n} h(p_{i,R})+h(p_{i,S})-h(p_{i,R}+p_{i,S}) \tag{3}$$

Among them, Pi, R represents the probability I of the first index in R, and h(x) = -xlogx.

3.2 Comparison of the parameter type

Algorithm 1 converts the operation parameters of the SAWSDL service to get the parameter element set of fine-grained measure, and the type of every element is the built-in type in XML Schema. The similarity function dataTypeSim is used to compare the similarity of the two basic data types. However, the similarity measure only considers a part of basic data types. This paper extends the similarity function in order to make it support the comparison of more data types. The data types supported by this method are shown in table 1. At the same time, we use theimilarity measurement method based on information loss to compare any two data types and the detailed similarity values are shown in Table 2.

Table 1 Division of the common data types

group	XML Schemadata types
Any Group	anyURI
Integer Group	integer, byte, short, long, unsignedLong, unsignedInd, Int, unsignedShort, unsignedByte, nonPositiveInteger, negativeInteger, nonNegativeInteger, positiveInteger
Real Group	float, double, decimal
String Group	String, normalizedString
Data Group	data, dataTime, duration, aDay, gMonth, aMonthDay, gYear, gYearMonth, time
Boolean Group	boolean

Table 2 The similarity value of type comparison

r vs. s	Any	Integer	Real	String	Date	Boolean
Any	1.0	1.0	1.0	1.0	1.0	1.0
Integer	1.0	1.0	0.5	0.3	0.1	0.1
Real	1.0	1/0	1.0	0.1	0.0	0.1
String	1.0	0.7	0.7	1.0	0.8	0.3
Data	1.0	0.1	0.0	0.1	1.0	0.0
Boolean	1.0	0.1	0.0	0.1	0.0	1.0

3.3 fine-grained parameters matching

According to the above all kinds of calculation results, the similarity between the two fine-grained parameters P γ and PS can be defined as follows:

$$\begin{aligned}&Sime_e(p_r,p_s)\\&=\frac{1}{\sum_{j-1}^{5}x_j}\cdot(x_1\cdot S_{name}(p_r\cdot pNmae,p_s\cdot pName)\\&+x_2\cdot S_{text}(p_r\cdot pDoc,p_s\cdot pDoc)\\&+x_3\cdot S_{\text{annotation}}(p_r\cdot p\text{Annotation},p_s\cdot p\text{Annotation})\\&+x_4\cdot S_{\text{type}}(p_r\cdot p\text{Type},p_s\cdot p\text{Type})\\&+x_5\cdot S_{\text{sc}}(p_r\cdot \text{SC},p_s\cdot \text{SC}))\end{aligned} \tag{4}$$

The formula shows that the similarity of two fine-grained parameters is composed of the name similarity, the text similarity, the semantic annotation the type similarity and the semantic context similarity, and Xi (i ∈{1,2,3,4,5}) is the weight of each part.

Finally, in calculation formula 1, the similarity between two parameters can be defined as : operation similarity 1 of two parameters in the 1 set can be defined as:

$$Sim_p(X,Y) = \frac{1}{|X|}\sum_{x\in X} \max_{y\in Y} Sime_e(x,y) \quad (5)$$

For each request parameters, from the matching parameters to find a parameter with the greatest degree of matching, and the matching value is in accordance with the matching degree of the request parameters. Finally, the average value of all request parameters is regarded as matching values as the similarity value of the request set and the matching parameters set.

EXPERIMENTAL EVALUATION

4.1 experimental data sets

The data set used in the experiment is mainly SAWSDL-TC2 data set for testing SAWSDL services. The reference matcher used in this paper is he various variants of SAWSDL-MX, including SAWSDL-WA, SAWSDL-M0, SAWSDL-M0+WA and SAWSDL-M1. The SAWSDL-Analyzer method with the help of SAWSDL-WA uses the structural similarity between the request and the service to sort candidate service through a recursive computation, and uses the node similarity to calculate the matching comparison of structural information involving the element name, the data type, the structure attribute, and the features of data instances. SAWSDL-M0 is a pure logical matcher based on the semantic annotation of SAWSDL operation parameters, and its logic matching can be divided into five grades: Exact, Plug-in, Subsume, Subsumed-by and Fail. According to the matching degree the candidate services will arrange them in a descending order: Exact>Plug-in>Subsumes>Subsumed-by>Fail, and services with the same semantic matching degree will be randomly sorted. On the basis of AWSDL-M0 matching, SAWSDL-M0+WA will sort the service of every semantic matching level according to the text similarity of the service interface semantic annotation.

We call the matcher put forward in the text as FGDSM (Fine-Grained Data Semantics based Matcher). At present, FGDSM has been added to the Web service matching device SAWSDL-iMacher, which make it become a convenient service matching mechanism. At the same time, this matcher has been configured to the evaluation environment SME2 of the general semantic Web service matcher. All experimental results in this paper are generated in the evaluation environment SME2.

4.2 experimental results and analysis

Table 3 summarizes the average accuracy proportion of every matcher in the test data set and the average query response time (AQRT). The results showed that the SAWSDL-WA matcher completely based on grammatical structure has a relatively low average accuracy rate, and the longest operation time. The average FGDSM has a higher average accuracy rate of 0.68, better than the SAWSDL-MX1 matcher based on machine learning, and because of avoiding the computational cost caused by structure comparison, its average response time is the shortest.

Table 3 the average accuracy proportion (MAP) and the average query response time (ARQT)

	FGDSM	SAWSDL-WA	SAWSDL-MX1	SAWSDL-M0+WA
MAP	0.69	0.32	0.67	0.62
ARQT	2.75s	11.94s	8.16s	15.47s

Figure 1 shows the matching effect of FGDSM and the matcher of description elements based on a single SAWSDL in SAWSDL-TC2. Among them, Name is the service name matcher based on SAWSDL, and it adopts the best similarity metric Dice's Codfficience to measure the similarity metric of service name. IO_IR is the semantic annotation matcher of service operation parameters based on SAWSDL, and it adopts Euclidean Distance to measure the similarity of semantic annotation sets. IO_Semantic is the semantic matcher of semantic annotation based on SAWSDL service operation parameters. The experiments show that, FGDSM is better than the matching method based on the service name and the logic matching method based on semantic annotation of operation parameters. When its recall rate is less than 20%, the average accuracy rate of FGDSM is slightly higher than that of IO_IR. If its recall rate point is greater than 20%, the average accuracy rate of FGDSM is less than IO_IR. The above experiments show that SAWSSDL service discovery based on fine-grained data semantic can not only

achieve better service discovery effect, but also avoid the high computational cost of structure matching, which can automatically, fast return the available service set in a relatively short period for the users. Therefore, this method can better satisfy the users' demand for service discovery.

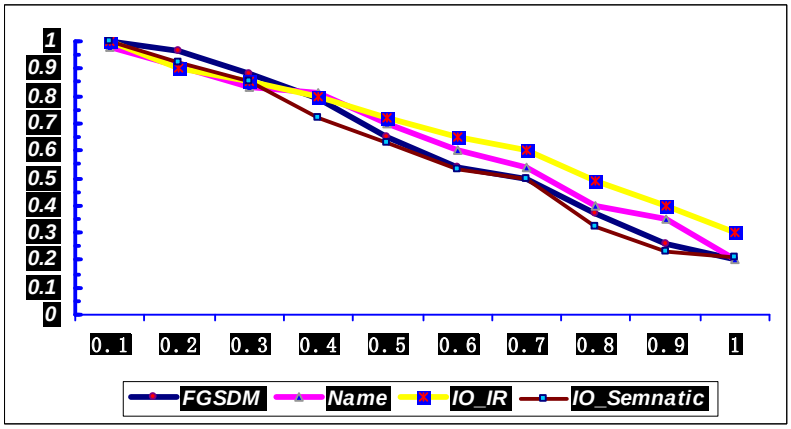

Figure 1 results of the data set SAWSDLTC2

CONCLUSION

For the deficiencies of current SAWSDL service operation parameter matching method, this paper proposes a service matching method SAWSDL based on fine-grained data semantic, and its core idea is to convert the message-level parameters of the complex data structure into many element sets with basic data types according to the structural features and semantic annotation features, which can make the fine-grained data parameters have such information as a name, a natural language description, semantic annotation and context and so on, and design a matching method of fine-grained parameters. Compared with the matching methods based on structure, the computational complexity of this method is greatly reduced, and the basic data semantic of SAWSDL service is maintained, which make the matching method of semantic Web service not dependent on specific labeling method. The experimental results show that, this method has a lower computational cost and a good retrieval performance.

REFERENCES

[1] Gelan Yang, Huixia Jin, and Na Bai. *Mathematical Problems in Engineering*, **2013**, vol. **2013**, Article ID 272567, doi:10.1155/2013/272567.
[2] YANG, Gelan, Yue WU, and Huixia JIN. *Journal of Computational Information Systems*, **2012**, 8(10): 4315-4322.
[3] Wu Yue, Gelan Yang, Huixia Jin, and Joseph P. Noonan. *Journal of Electronic Imaging*, **2012**, 21(1): 013014-1.
[4] Gelan Yang, Huixia Jin, and Na Bai. Mathematical Problems in Engineering, **2014**, vol. **2014**, Article ID 632060, 13 pages, 2014. doi:10.1155/2014/632060.
[5] Gao W., Y. Gao and L. Liang. *Journal of Chemical and Pharmaceutical Research*. **2013a**,5(9):592-598.
[6] Zhang B., Zhang S., Lu G.. *Journal of Chemical and Pharmaceutical Research*, **2013**, 5(9), 256-262.
[7] Mike P, Papazoglou, Dimitrios Georgakopoulos. *Service-Oriented Computing. Communications of the ACM*, **2003**, 46(10), 25-28.