Service Data Correlation Modeling and Its Application in Data-Driven Service Composition

Zhifeng Gu, *Student Member*, *IEEE*, Bin Xu, *Member*, *IEEE*, and Juanzi Li, *Member*, *IEEE*

Abstract—In this paper, we propose the Service Data Link model (SDL), a service relationship modeling schema, to describe service data correlations, which are data mappings among the input and output attributes of services. SDL recognizes the close correspondence between service data correlations and webpage hyperlinks, and defines service data correlations with explicit declarations, making it more expressive than the implicit method. We developed an XML implementation for SDL that can be seamlessly integrated into WSDL, the primary web services modeling language nowadays, and serves as an extension of metadata of services interfaces. An application of the SDL model in the domain of data-driven automatic service composition is then presented. First, we combine SDL with the Service Dependency Graph domain model developed by Liang, and present $\mathrm{SDG}+$, our enhanced model which extends the expressive power of SDG to include attribute quantifiers, attribute transforms, and explicit dependencies. Then, we show how $\mathrm{SDG}+$ can be used to improve the performance of composition algorithms in this domain.

Index Terms—Web services modeling, metadata of services interfaces, relationship modeling schema, extensions, service composition.

1 Introduction

Service-oriented Computing (SOC) has been widely accepted as an important new computing paradigm over the Internet. In recent years, research in SOC has shifted from description, publication, and discovery of a single service to composition and coordination among multiple services and peers. This increase in complexity necessitates new approaches that recognize and utilize various types of service relations. In this paper, we focus on a specific type of service relation, namely data correlation. We give a formalism for service data correlation and an application of our formalism in the domain of data-driven automatic service composition.

Fig. 1 gives an illustration of service data correlation. Suppose that there are two webpages. The first provides an interface for product list, i.e., a list of *ProductIDs* resulting from a search request. The other webpage provides an interface giving the details for a given *ProductID* from the product list. The data correlation between these two webpages is usually established by a hyperlink.[1]

Suppose that we wish to migrate our application to SOC by creating a service layer that provides the same functionality as these two webpages. A problem arises regarding how to express the corresponding data correlation between *Operation A* and *Operation B*. As far as we know, there is still no standardized solution to this problem. One possible way is to provide extra tags in the output document of *Operation A*. However, this approach has serious shortcomings: 1) for data models that allow extensibility, it may increase the verbosity of the output document and may introduce interoperability issues and 2) for strict data models that do not allow extensibility, the approach is infeasible unless the original data models are revised.

It is obvious that the data correlation between *Operation A* and *Operation B* does exist even though it is not explicitly defined. We may reestablish this data correlation by matching and reasoning about the I/O documents of *Operations A* and *B*. We refer to this as an implicit method of expressing service data correlations. However, the expressive power of the implicit method is limited in several important ways.

First, when expressing a collection of data correlations, the implicit method often introduces undesired or meaningless data correlations. This is explained in Fig. 2. In the figure, p_i is a service operation, a is an attribute, $p_i \rightarrow a$ means a is an output attribute of p_i, $a \rightarrow p_i$ means a is an input attribute of p_i, and $p_i \xrightarrow{a} p_j$ means a data correlation between p_i and p_j. Given the I/O definitions as shown in Fig. 2a, we can derive four data correlations as shown in Fig. 2b; however, the data correlations in Fig. 2b are highly interrelated. For example, we cannot derive $p_1 \xrightarrow{a} p_3$ without $p_2 \xrightarrow{a} p_3$. So it is usually unable to specify an arbitrary data correlation set precisely in an implicit way, for example, the set containing two data correlations as shown in Fig. 2c. Thus, the implicit method is a coarse-grained approach. A service operation producing attribute a will introduce a data correlation with every service operation consuming a; this is especially problematic with common attributes such as *name*, *address*, etc.

Second, the implicit method cannot effectively handle data correlations that need data transform, a time-consuming

1. Note that, here, the meaning of *hyperlink* is not limited to URLs, but generalized to HTTP GET/POST requests triggered by URLs or *input* fields in webpages.

• Z. Gu is with Shanghai Baosight Software Co., Ltd., No 515 Guoshoujing Rd, Zhangjiang Hi-tech Park, Pudong, Shanghai 201203, P.R. China. E-mail: guzhifeng@baosight.com.
• B. Xu and J. Li are with the Department of Computer Science and Technology, Tsinghua University, Room 10-206, East Main Building, Beijing 100084, P.R. China. E-mail: xubin@tsinghua.edu.cn, ljz@keg.cs.tsinghua.edu.cn.

Manuscript received 20 Nov. 2008; revised 17 July 2009; accepted 12 Apr. 2010; published online 26 May 2010.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2008-11-0105.
Digital Object Identifier no. 10.1109/TSC.2010.22.

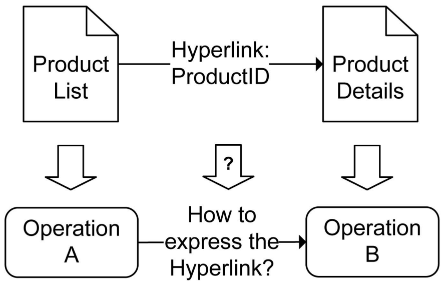


Fig. 1. How to express the corresponding data correlation at the service layer?

and error-prone task. Despite the existence of a number of commercial tools to aid in this process, the creation of data transforms requires dedicated input from domain experts and has not yet been fully automated. Moreover, transform from attribute a to a' is usually not unique, and in many cases, it is not true that the transforms from a to a' can be replaced with each other. For example, given $p \to a$, $a' \to p'$, and twodata transforms: $T_1 : a \mapsto a'$, $T_2 : a \mapsto a'$, we may derive two data correlations: $p \xrightarrow{T_1} p'$ and $p \xrightarrow{T_2} p'$. However, it is possible that T_1 is dedicated to some other service operations, and does not work well with p and p'. In such cases, we may want to suppress the undesired dependency; however, it is obvious that we cannot do this in an implicit way due to the lack of independence again.

Third, the implicit method is unable to distinguish between instance-level and definition-level data correlations. Fig. 3 illustrates the relationship between service definitions and service instances. Each service definition may have several implementations, and each implementation may have several instances; instances may share a common data source or use its own private data source. The implicit method assumes that the instance-level data correlations are always consistent with the definition-level data correlations, a situation that is not always true. It is possible that a definition-level data correlation fails to work when being applied to the instances of the definitions. There might be variety of reasons for such failures, and two major reasons are given below:

- **Broken conformance of implementations.** The specifications of real-world services might be very complicated, and are usually specified in natural languages, so it is hard to completely check the conformance of an implementation in a formal way.
- **Incompatible data sources.** The back-end data sources used by service instances may be incompatible across companies and organizations. For example, an *ID* generated by a service instance of company A is very likely to be invalid in the scope of the service instances of company B.

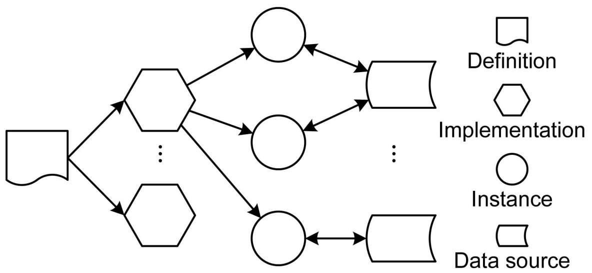


Fig. 3. Service definition and service instance.

Fourth, the implicit method is unable to assign extended information to a data correlation such as a priority rank or QoS properties. Moreover, the implicit method is based on an assumption that if the I/O attributes of two service operations are syntactically matched, the dataflow between them should work. However, in practice, this assumption is very likely violated due to interoperability issues introduced by real-world scenarios. In such cases, we may need to explicitly declare that the data correlations among some particular service operations should be avoided or enforced.

In order to overcome the problems associated with the implicit method of data correlation, we propose the Service Data Link model (SDL), which explicitly defines service data correlations. Our formal SDL model provides mechanisms to support data transforms, annotations, as well as decoupling of service data correlation definitions from data instances. We also formalize the Service Dependency Graph (SDG) model of Liang et al. [1], [2] and combine it with SDL to produce SDG+. This enhanced model provides a rich foundation for expressing the complex service data correlation relationships found in the domain of data-driven automatic service composition.

The rest of the paper is organized as follows: In Section 2, we give the definition of the SDL model. An XML implementation of SDL is presented in Section 3. From Section 4 to Section 6, we give an application of SDL in the domain of data-driven automatic service composition. In Section 4, we give a brief introduction to SDG, a domain model proposed by Liang and Su [2], and present a formalism of SDG. Then, in Section 5, we propose SDG+, a combination of SDL and SDG. We address the problems exposed in SDG, and give the corresponding solutions based on SDL. A formal definition of SDG+ is given at the end of this section. In Section 6, we show how SDG+ is used to

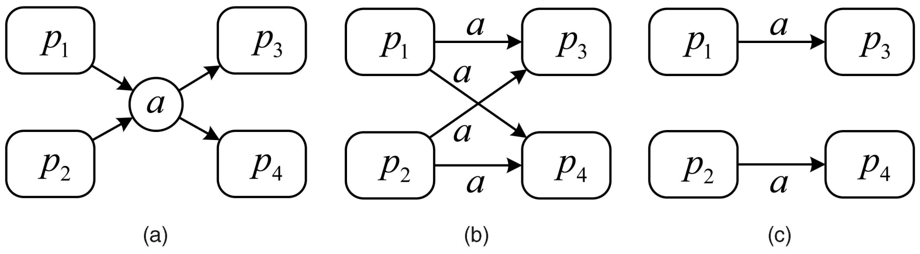


Fig. 2. Implied data correlations are interrelated.

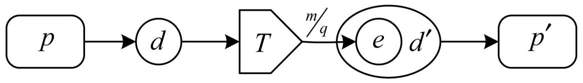


Fig. 4. An illustration of the SDL model.

improve the performance of our solution for WS-Challenge 2007. Finally, we discuss related works in Section 7, and draw some conclusions in Section 8.

2 The SDL Model

2.1 Definition

First, we give a formal definition of the SDL model. The following are the concepts used by SDL:

- d, d', d_i, d_I, d_O: documents. Each document is a data entity with arbitrary structures.
- p, p', p_i: service operations. Each service operation in SDL has one and only one document as its input, and one and only one document as its output. In other words, each service operation can be abstracted as a pair (d_I, d_O).
- e, e', e_i: elements. Each element is an addressable subcomponent of a document. The way to locate an element in a document depends on the implementation.
- q, q', q_i, m: quantifiers. We introduce the following widely accepted quantifiers:
  - 1 (one and only one),
  - $+$ (one or more),
  - $*$ (zero or more), and
  - ? (zero or one).
- T, T', T_i: transforms that map a document to an element. Given document d and element e, the transform between them is not supposed to be unique. We do not engage any implementation details of transforms. This is supposed to be handled in the research area of schema mapping [3], [4].
- A, A', A_i: annotations.

Based on these concepts, we give the definition of SDL.

Definition 1 (SDL). *An SDL is a quintuple (p, p', L, T, A), where:*

- *p is a service operation, which may refer to a definition or an instance.*
- *p' is a service operation, which may refer to a definition or an instance.*
- *$L : d' \rightsquigarrow e^q$ is a locator that addresses an element e in d', the input document of p'. q is the quantifier of e.*
- *$T : d \mapsto e^m$ is a transform that maps d, the output document of p, to an element e. m, a quantifier, is the multiplicity factor of T.*
- *A is annotations associated with this SDL. A is optional.*

The SDL quintuple can also be denoted by $T(p) \overset{A}{\Rightarrow} L(p')$, which is more compact and more intuitive. Fig. 4 gives an illustration of the SDL model. The locator L is not included directly in the figure, but the element e located by L is figured out to connect with the output of transform T. m and q are the quantifier of the output element of T and the located element of L, respectively.

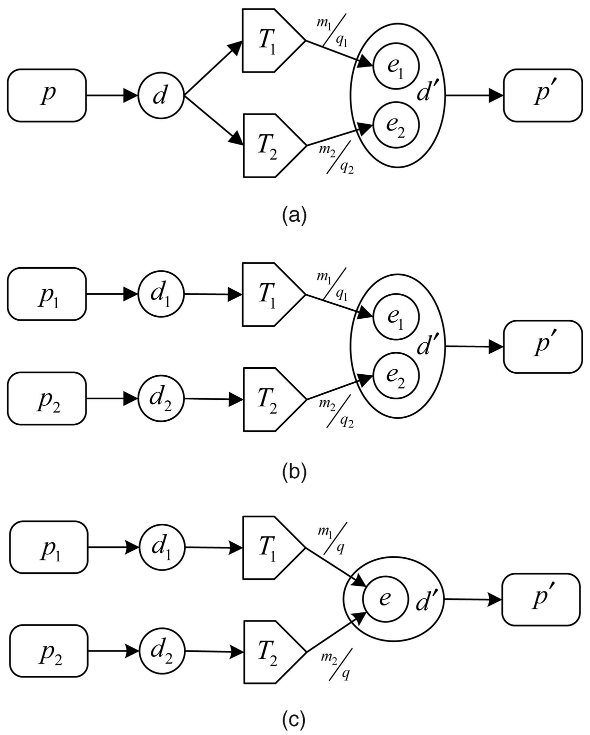


Fig. 5. Combination patterns of SDLs. (a) 1:1, (b) N:1, and (c) merge.

2.2 Combination of SDLs

It is obvious that given a service operation p', there might be many SDLs pointing to various elements within its input document d'. So there might be a number of SDL combinations to fulfill the elements in d'. Three possible combination patterns are shown in Figs. 5a, 5b, and 5c. However, SDL is simply a data correlation between two service operations; we do not assume that any SDL combinations are valid. Whether a combination of SDLs is valid is decided by high-level logic.

Moreover, unlike hyperlinks between two webpages, which may include user inputs, SDL combinations are usually *incomplete*, which means that the input document of one service operation usually needs inputs from the application layer, and cannot be fulfilled by any combination of SDLs. This is a significant difference between hyperlinks and service data correlations. In the XML implementation of SDL, we provide a simple mechanism to group a set of SDLs from p to p', and to specify whether the combination of the grouped SDLs is *complete* to p' (see Section 3.1).

In Fig. 6, we show a dataflow pattern that cannot be supported by the combination of SDLs. This is also an N:1

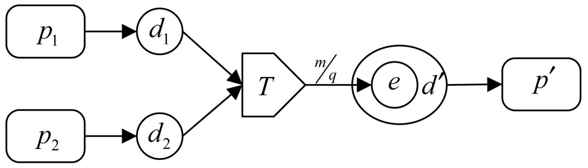


Fig. 6. An unsupported dataflow pattern.

pattern like that shown in Fig. 5b. This pattern is needed if e_1 and e_2 are interrelated, and cannot be generated independently by T_1 and T_2. Although such requirements really exist, we think that they are too application specific to be handled by general models like the SDL model.

3 Implementation of SDL

In this section, we introduce an XML implementation of the SDL model and its integration with WSDL.

3.1 Specifying SDL with XML

First, we developed an XML schema to describe the SDL model. As the XML schema is quite verbose, we use a comprehensive example to show how it works. The example SDL is intended to describe the data correlation illustrated in Fig. 1, and the detailed XML snippet is given as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<wsdatalink name="example"
          xml:ns="http://example.com/">
  <source> <!-- p -->
    <definition portType="ns:ServiceA"
           operation="getProductList"/>
  </source>
  <target> <!-- p' -->
    <definition portType="ns:ServiceB"
           operation="getProductDetails"/>
  </target>
  <!-- a group of data correlations
       from p to p' -->
  <mappings complete="true">
    <mapping>
      <!-- L -->
      <locator quantity="1" part="arg0">
        <xpath>/ProductID</xpath>
      </locator>
      <!-- T -->
      <transform multiplicity="*"
                 part="result">
        <xpath>/Products/ProductID</xpath>
      </transform>
    </mapping>
  </mappings>
  <!-- A: annotations -->
  <ext:weight>1</ext:weight>
  <ext:desc>A demo SDL</ext:desc>
</wsdatalink>
```

Some comprehensive comments have been added to this snippet to show how each piece of code corresponds to the elements in the SDL quintuple. Note that in the rest of this section, the meanings of the terms, "attribute" and "element," are supposed to be consistent with the meanings in the terminology of XML.

At the beginning of the XML snippet, two operations, p and p', are given. In this example, p and p' refer to two operation definitions in WSDL, each one is identified by two attributes: portType and operation. If p and p' refer to two instances, we also provide an element instance, which identifies an operation instance by three attributes: service, port, and operation.

The next part is a group of mapping rules. Each rule consists of two elements, a locator and a transform, which correspond to L and T in the definition of SDL, respectively. Locator may contain two attributes, quantity and part, both optional. Suppose $L : d' \rightsquigarrow e^q$, then quantity corresponds to the quantifier q. Part refers to a part in the input message definition of p'. If the message definition contains only one part, then the part attribute can be omitted. The enclosed XPATH expression is the core part of a locator. In this example, the XPATH expression selects the root element ProductID. Transform may also contain two optional attributes: multiplicity and part. Multiplicity is the quantifier of the output of this transform, and part refers to a part in the output message definition of p. In our schema definition, a transform can be implemented in three ways: XPATH, XSLT, and scripts (such as javascript, beanshell, etc.), which correspond to three possible subelements of transform: xpath, xslt, and script. In the example, the transform is implemented by a simple XPATH expression, which extracts a list of ProductIDs from the output document of p. Note that multiple mapping elements may be specified within one mappings group. This allows some particular semantics to be expressed on the combination of the enclosed mapping elements (i.e., SDLs), for example, the complete attribute of mappings, which indicates whether the combination of the grouped mapping elements is complete to construct the input document of p'. Obviously, in this example, the complete attribute should be true.

The final part is some extensibility elements (i.e., *xs:any*), which correspond to the annotation part of the SDL model. In principle, any valid XML snippets can be inserted here. In the example, we simply add a weight and a description to this SDL.

3.2 Embedding SDL into WSDL

As the XML representation of SDL contains references to entities defined in WSDL documents, such as messages and operations, it is a natural idea to embed SDLs into the WSDL documents that contain the referred entities. Fortunately, the extensibility of WSDL provides an easy way to achieve this integration.

A WSDL document can be logically divided into three parts [5]: XML schema, abstract description, and concrete description. The XML schema defines concepts/types in the domain and should be fully shared and reused. The abstract description defines *portType*, *operation*, and *message* based on imported XML schemas. The concrete description, containing definitions of *binding*, *service*, and *port*, describes service instances of the imported abstract descriptions. Obviously, the distinction between definition-level and instance-level SDLs is very well matched to this logical structure. Fig. 7 gives an illustration of our integration solution.

Technically, the W3C schema of WSDL [6] allows certain WSDL elements containing extensibility elements. However, the WS-I Basic Profile 1.1 defines more relaxed extensibility rules. That is, every WSDL element may have extensibility elements and extensibility attributes. So, in principle, SDLs can be put under any element of WSDL; however, in order to prevent confusion, definition-level SDLs are usually specified under *operation*, while instance-level SDLs are usually specified under *port*.

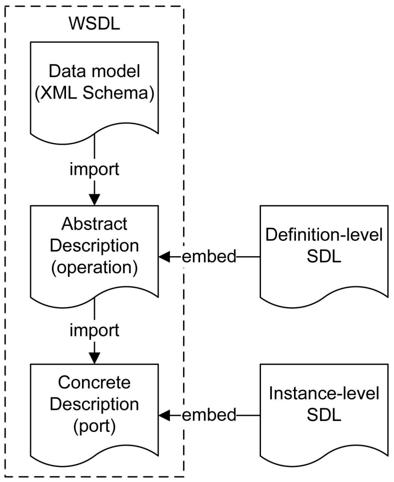


Fig. 7. Embed SDL into WSDL.

There might be two ways to embed SDL in WSDL. The first one is inline embedding. In this way, we put SDL declarations directly under WSDL elements. The other option is to specify SDL in external files, and then import these files into WSDL under appropriate elements.

3.3 Discussion

With this XML implementation of SDL, people of various roles are able to describe data correlations among web services according to their respective scopes, and share SDLs with others. For example, here, we are going to list three possible roles that may involve in the creation of SDLs. First, a service provider may create SDLs among his own services and business partners' services, and publish them together with service descriptions. Second, an independent organization may create and collect SDLs of his particular interests, and share SDLs freely or as a commercial service. Third, an end user of services (probably a service composer) may create and collect SDLs into his own database for future reference or sharing with his colleagues.

Regarding the issue of maintenance, SDLs are supposed to be created and maintained in a decentralized manner just as hyperlinks, so there will not be such problems that massive SDLs need to be maintained at anyone side. For example, a service provider only needs to maintain the SDLs published by himself, which obviously will not be a huge cost.

4 The SDG Model

Beginning from this section, we will present an application of SDL in the domain of data-driven automatic service composition. Research in service composition covers a variety of topics, among which automatic composition is a major branch. Automatic service composition (a.k.a. service synthesis) aims to create service compositions that can satisfy given constrains such as temporal behaviors [7], [8], pre/postconditions [9], [10], and input/output attributes [2], [11]. We particularly focus on a class of automatic service composition problems in which each service operation is modeled as a black box with a set input attributes and a set of output attributes, and the composition task is required to generate a composite service that produces the set of desired attributes by consuming the given attributes.

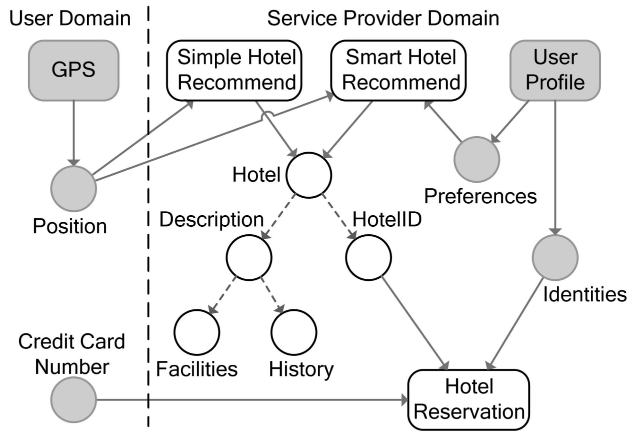


Fig. 8. An intuitive example of SDG.

The composition task solely focuses on matchmaking of service I/O attributes, but does not take other factors, such as data semantics and functional semantics, into account; therefore, we use the term "data-driven" to capture the characteristic of such kind of service composition problems.

SDG, proposed by Liang and Su [2], [1], is a domain model for data-driven automatic service composition problems. Our application first combines SDL with SDG, and presents a new proposal, the SDG+ model, which extends the expressive power of SDG to include attribute quantifiers, attribute transforms, and explicit dependencies. Then we make use of SDG+ to improve the performance of our composition solution for WS-Challenge 2007. In the rest of this section, we give an introduction to SDG and present a formalism of this model. Details of the SDG+ model and its application in WS-Challenge will be given in the next two sections.

4.1 Introduction to SDG

SDG is an AND/OR graph that shows all the possible input-output dependencies among different services. Within SDG, services are modeled as operation nodes with an input set and an output set, which contain attributes as their elements. Attributes are abstraction of data entities/objects. An attribute can be a simple attribute or a composite attribute. Composite attribute is composed of a set of simple attributes and composite attributes, while simple attribute is logically atomic [2].

Fig. 8 gives an example of SDG.[2] In the figure, the attribute node, represented by a circle, is OR node, which means that all the directed edges connected to it are logically ORed. The operation node, represented by a rectangle, is AND node, which means that all the directed edges connected to it are logically ANDed. An intuitive explanation is that all the input attributes have to be satisfied before the operation can be invoked. On the other hand, a particular attribute node can be produced by any operation node that has the attribute in its output set.

Regarding composite attribute, it can be decomposed into subattributes, which can then be fed into operation nodes. However, there is no such an assumption that the

2. Whether a shape is filled or not does not have any special meanings. The unfilled subgraph simply serves as a prototype of the abstract SDG shown in Fig. 10a.

aggregation of all the subattributes is semantically equal to the composite attribute, so a composite attribute cannot be solved by solving all of its subattributes. For example, in Fig. 8, `Hotel` cannot be solved by solving `HotelID` and `Description`.

In SDG, a "dependency" is a directed path between two operation nodes, for example, the path from `Simple Hotel Recommend` to `Hotel` to `HotelID` to `Hotel Reservation` in Fig. 8. It is obvious that a dependency is essentially a data correlation between two operations, and has a similar logical structure to SDL.

Given a set of known attributes and a set of required attributes, Liang has introduced a search algorithm to construct composite service templates, which is a subgraph of the AND/OR graph. The search algorithm starts from the *starting node*, a virtual AND node that is connected with all the required attribute nodes. The algorithm terminates at the *termination node*, another virtual AND node that connects to all the known attribute nodes and is considered to be solved.

Based on the SDG model, Liang proposed a semiautomatic service composition method, which is an iterative procedure consisting of two phases: 1) an automatic search algorithm to find composition candidates and 2) human evaluation. The search algorithm tries to find a solution graph with minimal cost (a minimal number of operation nodes and data nodes). When a solution graph is found, it is presented to the requester for evaluation. If the requester rejects the solution, the search algorithm will be applied again to find another solution. If no solution graph can be found or the requester rejects all the solutions, some "to-be-explored" operation nodes, which produce the required attributes directly or indirectly, will be added. Then, the search algorithm can be applied again.

4.2 A Formalism of SDG

In this section, we give a formal definition of SDG to make further discussion rigorous and clear. First, we introduce the following concepts in SDG:

- a, a', a_i: attributes.
- p, p', p_i: service operations. Unlike service operation in SDL, here, the input/output of a service operation is a set of attributes instead of a document. We say that a set of attributes is a special form of a document that consists of elements. The difference is that a document is not supposed to have a combinative structure, while a set of attributes is combinative to automatic composition algorithms.
- $a \rightarrow a'$: attribute relation, which means that a' is a direct subattribute of a.
- $p \rightarrow a$: output relation, which means that p produces a.
- $a \rightarrow p$: input relation, which means that p consumes a.

Obviously, an SDG is composed of relations including $a \rightarrow a'$, $p \rightarrow a$, and $a \rightarrow p$, so we give the definition of SDG as follows:

Definition 2 (SDG). *SDG is a triple (R_D, R_O, R_I), where:*

- $R_D = \{(a, a') \mid a \rightarrow a'\}$.
- $R_O = \{(p, a) \mid p \rightarrow a\}$.
- $R_I = \{(a, p) \mid a \rightarrow p\}$.

This definition abstracts an SDG as three sets of binary relations in which R_D is an abstraction to the data model, while R_O and R_I are abstractions to the definitions of service I/O. Next, we give the definition of dependency in SDG.

Definition 3 (Dependency). *A dependency in a given SDG (R_D, R_O, R_I) is a triple (p, p', Φ), where:*

- *Φ is a sequence of attributes $a_1, a_2, \ldots, a_{n-1}, a_n$ in which $(a_i, a_{i+1}) \in R_D, 1 \leq i < n-1$.*
- *$(p, a_1) \in R_O$.*
- *$(a_n, p') \in R_I$.*

Intuitively, dependency (p, p', Φ) is a directed path in SDG from p to p', which looks like $p \rightarrow a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_{n-1} \rightarrow a_n \rightarrow p'$. Usually, we denote dependency in a more intuitive and compact form: $p \xrightarrow{\Phi} p'$.

According to Definitions 2 and 3, given an SDG G, we may derive a set of dependencies, denoted by $\Delta(G)$. This is an implicit way to express dependencies (data correlations). Its expressive power is restricted according to the discussions given in Section 1. Particularly, in the domain of data-driven automatic service composition, the fourth restriction, regarding heuristic information associated with data correlations, is the most critical.

5 The SDG+ Model

Although SDG is powerful enough to model a variety of data-driven service composition problems such as WS-Challenge, we argue that it is not perfect yet and may be improved on several aspects to handle more complicated requirements. In this section, we figure out three problems exposed in SDG, and propose our solution extensions, respectively, which are attribute quantifier, attribute transform, and explicit dependency. Finally, we conclude the SDG+ model from the proposed extensions.

5.1 Attribute Quantifier

The first problem is that SDG does not provide a mechanism to specify the quantity of an attribute. In SDG, there are two types of attributes: simple attribute and composite attribute. Simple attribute is a data entity/object that has a system predefined primitive data type, while composite attribute is composed of a set of simple and composite attributes [2]. Obviously, this simple metadata model misses something to express the quantity of an attribute. Suppose that there are two service operations: One provides a hotel search service that returns a list of `Hotel` attributes and the other one provides a hotel reservation service that takes a `Hotel` attribute as the input. In this case, the relationship between the `Hotel` attribute and the list of `Hotel` attributes is supposed to be irrelevant. As a result, if there is not a service operation that takes the list as its input and returns one `Hotel` attribute from the list, these two service operations are unable to be composed, although actually they are highly interrelated.

Referring to the quantifiers introduced in SDL, we may specify a quantifier q to describe the quantity of the I/O attributes of a service operation, e.g., $p \rightarrow a^q$ and $a^q \rightarrow p$. Also, each subattribute can be associated with a quantifier,

TABLE 1
Operation Rules of $q \otimes q'$

q \ q'	1	$?$	$*$	$+$
1	1	$?$	$*$	$+$
$?$	$?$	$?$	$*$	$*$
$*$	$*$	$*$	$*$	$*$
$+$	$+$	$*$	$*$	$+$

e.g., $a \rightarrow a'^q$. If no quantifier is specified, the default quantifier is 1.

With attribute quantifier, the relationship between `Hotel`, an attribute, and `Hotel`*, a list of `Hotel` attributes, can be clearly addressed. As a result, dependencies between service operations producing `Hotel`* and service operations consuming `Hotel` will be exploited, while this is irrealizable in SDG.

After introducing attribute quantifier, we need to consider the issues about matchmaking on quantifiers. Before that, we need to define an operation on quantifiers, $\otimes : Q \times Q \mapsto Q$, $Q = \{1, ?, *, +\}$. The semantics of this operation is that given a^q and $a \rightarrow a'^{q'}$, the quantity of a' is $q \otimes q'$. Obviously, the high boundary and low boundary of $q \otimes q'$ are easy to calculate. For example, regarding $* \otimes ?$, the low boundary is 0, the high boundary is ∞, so $* \otimes ? = *$. Detailed rules of this operation are given in Table 1. It can be proved that this operation is commutative and associative by exhaustive enumeration.

Having this operation defined, we are ready to discuss two issues on matching attribute quantifiers. First, we need to compute the quantifiers of the subattributes produced by a service operation. For example, given a dependency $p \xrightarrow{\Phi} p'$ in SDG, if we extend it with attribute quantifiers, then each attribute on the dependency path will have a quantifier associated with it: $p \rightarrow a_1^{q_1} \rightarrow a_2^{q_2} \rightarrow \cdots \rightarrow a_{n-1}^{q_{n-1}} \rightarrow a_n^{q_n} \rightarrow p'$. Referring to the semantics of the quantifier operation, it is easy to see that the quantifier of a_n according to this path is $q_1 \otimes q_2 \otimes \ldots \otimes q_{n-1} \otimes q_n$. We denote this indirect output relation by $p \xrightarrow{\Phi} a_n^{q_1 \otimes q_2 \otimes \ldots \otimes q_{n-1} \otimes q_n}$. Furthermore, as the quantifier operation is associative, the quantifier of a_n can be specified in another form $q_1 \otimes q_\Phi$ in which $q_\Phi = q_2 \otimes q_3 \otimes \ldots \otimes q_{n-1} \otimes q_n$. q_Φ is a multiplicity factor of Φ, if we treat Φ as a special form of attribute transform (refer to Section 5.2).

Second, given $p \xrightarrow{\Phi} a^q$ and $a^{q'} \rightarrow p'$, we need to decide whether q and q' can be matched. Unfortunately, this is usually an application-specific issue. There are no general rules to make decisions. In this paper, we use a Boolean function $M(q, q')$ to define quantifier matching policies. For example, a simple policy may be defined as

$$M(q, q') = \begin{cases} true & \text{if } q = q', \\ false & \text{if } q \neq q'. \end{cases}$$

If we require that the quantifier of the provided attribute is "larger" than the quantifier of the required attribute according to partial order $+ > 1 > * > ?$, then the matching policy may be defined as

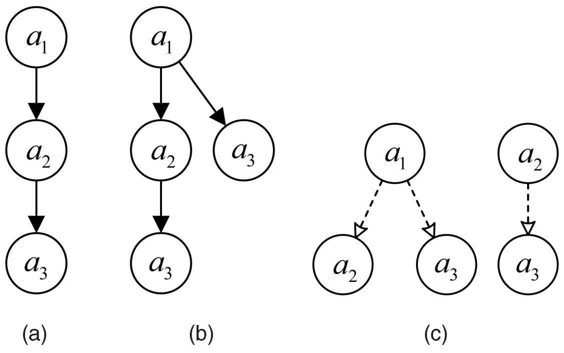


Fig. 9. Structural information is unnecessary.

$$M(q, q') = \begin{cases} true & \text{if } (q = +) \vee \\ & (q = 1 \wedge q' \neq +) \vee \\ & (q = * \wedge q' \neq + \wedge q' \neq 1) \vee \\ & (q = \ ? \wedge q' = \ ?), \\ false, & \text{otherwise.} \end{cases}$$

5.2 Attribute Transform

The second problem of SDG is about its modeling method of relations on attributes, and can be explained from two aspects.

First, SDG recognizes the simple *subattribute-of* relation between two attributes by introducing a hierarchical metadata model; however, we argue that the hierarchical structure of an attribute is actually unnecessary to SDG-based composition algorithms. Consider the example shown in Fig. 9. According to the semantics of SDG, the structure of attribute a_1 shown in Fig. 9a is logically equal to the version shown in Fig. 9b, in which attribute a_1 has an additional direct subattribute a_3, which is also an indirect subattribute of a_1. In fact, SDG only concerns whether an attribute is contained in another one, no matter it is directly or indirectly contained. So, the attribute relations shown in Fig. 9c are essentially required by SDG-based composition algorithms.

Second, considering the complexity of data model and data heterogeneity in real-world scenarios, the underlying mapping rules between two attributes may be very complicated. Although the *subattribute-of* relation may logically serve as an abstract form to express attribute relations like "attribute a_2 can be retrieved or constructed from a_1," it cannot clearly address noninclusion relations between two attributes.

To solve this problem, referring the concept of transform in SDL, we introduce the second extension to SDG, attribute transform. In our perspective, each attribute is treated as a logically independent data entity, and the relations on attributes are built through attribute transform, which is defined as follows:

Definition 4 (Attribute Transform). *An attribute transform is a mapping $T : a \mapsto a'^m$, where:*

- *a is the input attribute of the transform.*
- *a' is the output attribute of the transform.*
- *m, a quantifier, is the multiplicity factor of the transform. The default value of m is 1.*

Attribute transform is a similar concept to "transform" in SDL, but the input/output objects are different. The input and output of "transform" are document and element, respectively, while the input and output of attribute transform are both attribute. Considering that "attribute" and "element" are parallel concepts, it is obvious to see that attribute transform is a special form of transform, as attribute transform can only express relations between two attributes, while "transform" can express relations between a set of attributes (including one attribute) and another attribute. Again, like "transform" in SDL, we do not assume the uniqueness of attribute transform either. For example, in Fig. 9b, there are two paths (Φ) from a_1 to a_3, which may correspond to two attribute transforms.

It is a natural idea that given $T_1 : a_1 \mapsto a_2$ and $T_2 : a_2 \mapsto a_3$, we can derive another attribute transform by function composition: $T = T_2 \circ T_1 : a_1 \mapsto a_3$. However, we do not intend to support composition of attribute transform in our extended model because this will lead to hierarchical structures among attributes again, and introduce additional complexity into the model. Our standpoint is that all the detailed issues about attribute transform, including creation, composition, etc., should be shifted to works dedicated to these topics, while we simply assume that those works will generate a set containing all the possible transforms over the given attributes.

Given $p \rightarrow a^q$ and $T : a \mapsto a'^m$, we say that p can produce a' indirectly by mapping its output attribute a to a'. And according to the semantics and rules of quantifier operation defined in Section 5.1, the quantifier of a', when being produced by p, is $q \otimes m$. We denote this indirect output relation by $p \xrightarrow{T} a'^{q\otimes m}$, which can then be used to construct dependencies by matching input relations such as $a'^{q\otimes m} \rightarrow p'$. But before that, a quantifier matching policy will need to be defined first.

With attribute transform, the structural information of attributes can be dropped, as the relations on attributes can now be modeled by attribute transform instead of the hierarchical structure. At the same time, the assumption of "integrated ontology space" is no longer needed, as attribute transform enables mappings on heterogeneous data in principle.

Also note that the sequence Φ in a dependency (see Definition 3), which extracts subattributes from a given attribute, is a special form of attribute transform. And the quantifier q_Φ we have computed in Section 5.1 just corresponds to the multiplicity factor of attribute transform.

5.3 Explicit Dependency

The third problem is that SDG specifies dependencies in an implicit way, and its express power is restricted (refer to Section 4.2). To solve this problem, we propose the concept of explicit dependency to define dependencies with explicit declarations like SDL.

Definition 5 (Explicit Dependency). *An explicit dependency is a quadruple (p, p′, T, w), where:*

- *$T : a \rightarrow a'^m$ is an attribute transform.*
- *p is a service operation, $p \rightarrow a^q$.*
- *p′ is a service operation, $a'^{q'} \rightarrow p'$.*
- *w is the weights associated with this dependency.*

The explicit dependency quadruple can also be denoted by $p \underset{A}{\overset{T}{\Rightarrow}} p'$. It is obvious that explicit dependency is a special form of the SDL model. This can be explained from the following aspects:

1. Attribute transform is a special form of "transform" in SDL.
2. Service operation particularly refers to service definition in SDG.
3. The weight w is a special form of annotation.
4. SDG models service input/output as a set of attributes, so the locator L in SDL has no counterpart in the definition of explicit dependency.

5.4 Definition of SDG+

So far we have introduced three extensions to the SDG model, which are attribute quantifier, attribute transform, and explicit dependency. In this section, we will give a formal definition of SDG+ and a comprehensive comparison of SDG and SDG+.

First, we have the following concepts in SDG+:

- a, a', a_i: attributes.
- p, p', p_i: service operations.
- q, q', q_i, m: quantifiers.
- T, T', T_i: attribute transforms.
- A, A', A_i: annotations.
- $p \underset{A}{\overset{T}{\Rightarrow}} p'$: explicit dependency.
- $p \rightarrow a^q$: output relation, which means that p produces a, and the quantifier is q.
- $a^q \rightarrow p$: input relation, which means that p consumes a, and the quantifier is q.

Based on these concepts, we give the definition of SDG+.

Definition 6 (SDG+). *An SDG+ is a quintuple (R_O, R_I, Θ, Δ, M), where:*

- $R_O = \{(p, a, q) \mid p \rightarrow a^q\}$.
- $R_I = \{(a, p, q) \mid a^q \rightarrow p\}$.
- *Θ is a set of attribute transforms.*
- *Δ is a set of explicit dependencies.*
- *M is a quantifier matching function.*

Referring to the definition of SDG (Definition 2), SDG+ introduces three new items: 1) a set of attribute transforms Θ, which essentially corresponds to the relation set R_D of SDG, 2) a set of explicit dependencies Δ, and 3) a quantifier matching function M that is used to derive dependencies.

Similar to dependency derivation in SDG, we can also derive dependencies from an SDG+. We use the term "dependency+" to indicate the dependencies derived from SDG+.

Definition 7 (Dependency+). *A dependency+ in a given SDG+ (R_O, R_I, Θ, Δ, M) is a triple (p, p′, T), where:*

- $T \in \Theta$, $T : a \mapsto a'^m$.
- $(p, a, q) \in R_O$.
- $(a', p', q') \in R_I$.
- $M(q \otimes m, q') = true$.

The dependency+ triple can also be denoted by $p \xrightarrow{T} p'$, which formally looks like $p \xrightarrow{\Phi} p'$, a dependency in SDG.

TABLE 2
Comparison of SDG and SDG+

Level	Items	SDG	SDG+
Basic	Attribute Relation	$a \rightarrow a'$	$T : a \mapsto a'^m$
	Output Relation	$p \rightarrow a$	$p \rightarrow a^q$
	Input Relation	$a \rightarrow p$	$a^q \rightarrow p$
	Explicit Dependency	N/A	$p \overset{T}{\underset{A}{\Rightarrow}} p'$
Derived	Implied Dependency	$p \xrightarrow{\Phi} p'$	$p \xrightarrow{T} p'$
	Indirect Output Relation	$p \xrightarrow{\Phi} a$	$p \xrightarrow{T} a^{q \otimes m}$

A comprehensive comparison of SDG and SDG+ is given in Table 2. On the whole, the comparison is categorized into two levels: 1) basic level, which includes the basic items in SDG and SDG+, and 2) derived level, which includes the items that can be derived from the basic items.

Fig. 10 gives a visual comparison of SDG and SDG+. The SDG shown in Fig. 10a is an abstract form of the unfilled subgraph of Fig. 8, and its corresponding SDG+ representation is given in Fig. 10b in which the hierarchical structure of a_1 is converted into attribute transforms, and the input/output edges of operation nodes are all associated with a quantifier. Moreover, we add an explicit dependency $p_1 \overset{T_2}{\underset{A}{\Rightarrow}} p_3$ to show how to override the implied dependency+, particularly $p_1 \xrightarrow{T_2} p_3$ in this case.

6 Solution for WS-Challenge 2007

In this section, we give an application of SDG+, which takes advantage of explicit dependency in SDG+ to improve the performance of our solution for WS-Challenge 2007.

6.1 Introduction to WS-Challenge

WS-Challenge[3] is a competition of automatic service composition organized by annual conference of the IEEE CEC/EEE. The composition problem defined by WS-Challenge requires the inputs of a service operation to be satisfied by the outputs of some other service operations (see Section 6.2 for details), which is a typical data-driven service composition problem. As far as we know, WS-Challenge is the first attempt in the research community of web services to establish a benchmark on automatic service composition.

WS-Challenge defines two kinds of composition tasks: syntactic composition and semantic composition. Syntactic composition matches service I/O by attribute names (the *name* attribute of the *part* element), while semantic composition matches service I/O by attribute types (the *type* attribute of the *part* element). Both of them can be well modeled by SDG. Adopting SDG, we developed a composition algorithm for WS-Challenge 2006 [12], which won the championship of performance[4] in syntactic composition and the fourth place for performance of semantic composition. Of course, there are various methods to solve the WS-Challenge problem [13], [14], [15], [16], [17], [18]. Using SDG is just one of the possible solutions.

3. http://www.ws-challenge.org.
4. WS-Challenge ranks a solver on two aspects: performance and architecture.

As discussed above, the composition problem defined by WS-Challenge can be well modeled by SDG, as the data sets of WS-Challenge are quite simple, and there are no requirements to use attribute quantifier, attribute transform, or explicit dependency. However, we also noticed that according to a given data set, which corresponds to an SDG G, the set of the implied dependencies, $\Delta(G)$, is fixed. It means that the result of the dependency construction procedure can be reused across every composition request on this data set. This is the key idea of our improved solution. In fact, this is the origin of the idea of explicit dependency and even SDL, although explicit dependency is just a serialization of $\Delta(G)$ in this application.

6.2 Problem Definition

The goal of WS-Challenge is to find all the composition solutions as quickly as possible. Each composition solution is a chain of service operations as shown in Fig. 11. If the input and output sets given in the composition request are denoted by I_{req} and O_{req}, respectively, and the service operations in the chain are denoted by p_k, $1 \leq k \leq n$, and the input and output sets of p_k are denoted by I_k and O_k, respectively, then for syntactic composition, the following conditions must hold for the chain: 1) $I_k \subseteq O_{k-1} \cup I_{req}$, $1 < k \leq n$, 2) $I_1 \subseteq I_{req}$, and 3) $O_n \supseteq O_{req}$.

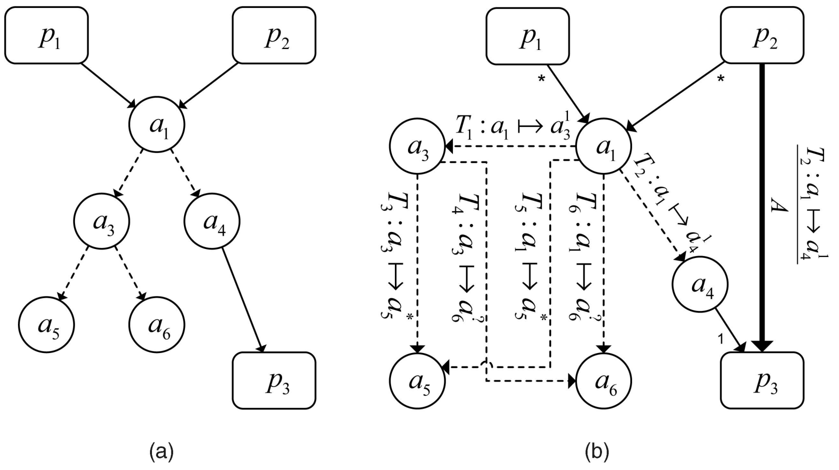
For semantic composition, the *subattribute-of* relation needs to be handled additionally. Given X, a set of attributes, we may derive a new set that contains X and all the direct and indirect subattributes of the elements in X. If such a set is denoted by $\widehat{X}$, then for semantic composition, the following conditions must hold for the chain: 1) $I_k \subseteq \widehat{O}_{k-1} \cup \widehat{I}_{req}$, $1 < k \leq n$, 2) $I_1 \subseteq \widehat{I}_{req}$, and 3) $\widehat{O}_n \supseteq O_{req}$.

6.3 The Baseline Solution

We use our program for WS-Challenge 2006 as the baseline solution. The overall flow of this solution is shown in Fig. 12 in which all the services are indexed by a hash map, Map_O, which associates an attribute with all the service operations that can produce the attribute. Note that the service operations that can produce the compatible attributes of the given attribute (a.k.a. its superattributes) are not included in the associated data; instead, these services may be retrieved by lookup Map_D, another hash map that associates an attribute with its direct superattributes.

For each service operation p in the service chain, as shown in Fig. 11, the *dependency construction* module first constructs dependencies in the form of $p' \xrightarrow{a} p$ by lookup Map_t against a, one input attribute of p, and also against the superattributes of a. Then the set of dependencies that goes through a can be retrieved as a union of all the returned results. Finally, the *composition algorithm* module computes the precedents of p by intersecting the sets of dependencies that go through each of its input attributes (the attributes contained in the given I_{req} are ignored). In this way, the service operation chain can be constructed iteratively.

Note that the *dependency construction* module is actually a part of the *composition algorithm* in our solution for WS-Challenge 2006. However, in order to show the differences between the 2006 solution and the 2007 solution, this conceptual module is abstracted. This module is the key to our improved solution for WS-Challenge 2007, as the



(a)
(b)
$T_1 : a_1 \mapsto a_3^1$
$T_2 : a_1 \mapsto a_4^1$
$T_3 : a_3 \mapsto a_5^*$
$T_4 : a_3 \mapsto a_6^?$
$T_5 : a_1 \mapsto a_5^*$
$T_6 : a_1 \mapsto a_6^?$
A

Fig. 10. A visual comparison of (a) SDG and (b) SDG+.

output of this module can be reused across every composition request on the same data set, and time consumption of this module is remarkable according to the whole flow.

6.4 The Improved Solution

The improved solution for WS-Challenge 2007 is shown together with the baseline solution in Fig. 12. In detail, we developed a preprocessing procedure that constructs dependencies for each service operation and saves them onto an on-disk database for reuse. In this way, the *dependency construction* module is bypassed by lookup another hash map, Map_Δ, which takes service operation as its key, and the data associated with the key are a set of lists, where each list contains the dependencies that go through a particular input attribute of the key.

After running the preprocessing procedure, the *composition algorithm* module may retrieve dependencies directly from Map_Δ, so according to a given request, the time cost on dependency construction, denoted by t_c, is eliminated,

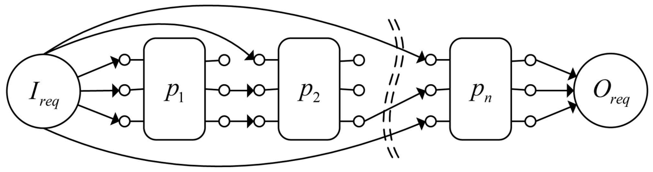
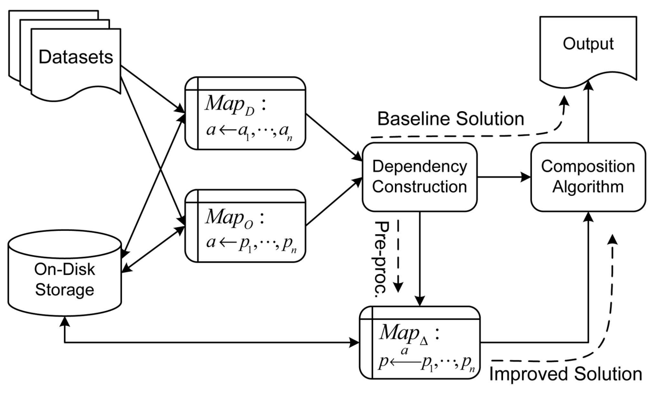


Fig. 11. A chain of service operations.



Datasets
Map_D : $a \leftarrow a_1, \cdots, a_n$
Map_O : $a \leftarrow p_1, \cdots, p_n$
On-Disk Storage
Baseline Solution
Output
Dependency Construction
Composition Algorithm
Pre-proc.
Map_Δ : $p \xleftarrow{a} p_1, \cdots, p_n$
Improved Solution

Fig. 12. Baseline solution and improved solution.

while at the same time, the additional cost of lookup Map_Δ, denoted by t_l, is introduced. On the whole, the overall/effective time saving, denoted by t_e, is $t_c - t_l$. Assume that the time costs of the baseline solution and the improved solution spent on processing the given request are t_b and t_o, respectively, then we have $t_b - t_o = t_e = t_c - t_l$. It is reasonable to evaluate the improved solution by an *effectiveness* factor λ, defined as follows:

$$\lambda = \frac{t_e}{t_b} = 1 - \frac{t_o}{t_b}.$$

Clearly, larger λ indicates better results.

Moreover, two additional time costs need to be considered in the improved solution.

The first is the time cost of the preprocessing procedure, denoted by t_p. Although it is possible that $t_e > t_p$ (e.g., request No. 11 in Fig. 13d), in most cases, t_p will be larger than $\overline{t_e}$, the average of t_e, because the preprocessing procedure needs to construct all the possible dependencies, not to mention other costs such as the creation of Map_Δ. Fortunately, the preprocessing procedure runs only once for each data set. Once preprocessing is finished, all the subsequent requests on the data set will be benefited, and hopefully, the preprocessing cost will be compensated after $\lceil t_p/\overline{t_e} \rceil$ runs. According to the competition rules of WS-Challenge, each set of requests will be run five times, so this improved solution is valid on this standpoint.

The second is the additional overhead of disk I/O for serializing and deserializing Map_Δ. Like the preprocessing cost, the cost of serializing Map_Δ occurs only once for each data set. However, the cost of deserializing Map_Δ occurs every time the composition program is started and initialized. If the composition program runs as a daemon, then this cost can also be compensated with the increase of the number of composition requests.[5]

Strictly speaking, the preconstructed dependencies are not explicit dependencies, as they are simply the serialization of all the implied dependencies of the given data set and do not have well adjusted weights. However, as they

5. Our program for WS-Challenge 2007 does not work as a daemon, so every time it is started, the time costs on deserializing Map_Δ will bring some negative impacts to the results. We will fix this problem in future versions.

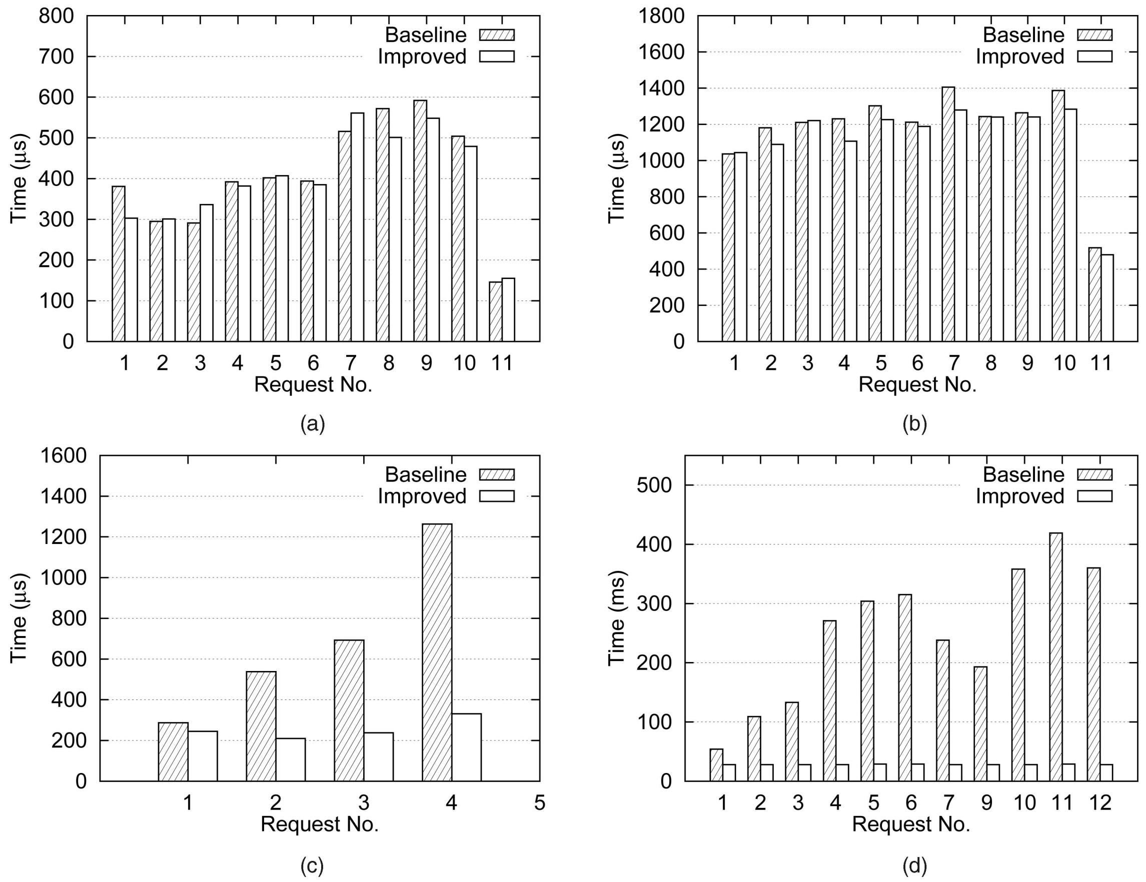


Fig. 13. Experiment results. (a) #1: composition1-20-32. (b) #2: composition2-100-32. (c) #3: composition_config_small. (d) #4: composition_config_large.

TABLE 3
Data Set Information

ID	Name	Services	Attributes[a]	Sub-attr. Rel.[b]	Ave. Inputs[c]	Preprocessing
1	composition1-20-32	2156	1580	0	33.49	25694 μs
2	composition2-100-32	8356	3060	0	33.49	92341 μs
3	composition_config_small	118	1590	1560	2.31	3918 μs
4	composition_config_large	978	979740	979650	3.00	298 ms

[a] For dataset #1 and #2, this column refers to the number of attribute names; for dataset #3 and #4, this column refers to the number of attribute types. Refer to Section 6.1 for more details.
[b] The number of *sub-attribute-of* relations in the dataset, which corresponds to the number of type inheritances.
[c] The average number of input attributes per service.

have been specified and stored in an explicit way, we think that their potential of expressiveness has been fundamentally expanded.

6.5 Experiments

We use four data sets from previous WS-Challenge competitions to evaluate the improved solution. The hardware platform is Celeron 1 GB, 512 MB RAM, and the software environment is Debian Linux (Sarge[6]). All timings are retrieved through the *gettimeofday* API.

Information about the four data sets is shown in Table 3. The first two data sets come from WS-Challenge 2005, while the last two data sets come from WS-Challenge 2006. WS-Challenge 2005 does not support type inheritance in XML Schema, so the values of the "Sub-attr. Rel." column of the first two data sets are zero. The preprocessing time cost of each data set is given in the last column, and each is an average of five runs.

The experiment results are shown in Fig. 13. All the time values are counted after program initialization has been finished, which means that the maps discussed in Sections 6.3 and 6.4 have been constructed or have been deserialized into the memory. We can see that there is no significant improvement on performance for the first two data sets, while for the last two data sets from WS-Challenge 2006, the improvement is quite impressive.

As the data sets from WS-Challenge 2005 do not support type inheritance, the *dependency construction* module only needs to lookup Map_O one time for each input attribute of a service. The cost is nearly the same as directly retrieving preconstructed dependencies, so t_e is very small, sometimes

6. http://www.debian.org/releases/sarge.

even negative. Simply speaking, the improved solution is totally inapplicable to the data sets of WS-Challenge 2005.

However, the data sets from WS-Challenge 2006 support type inheritance. Given an attribute, the *dependency construction* module needs to lookup not only the services that can produce the given attribute, but also all the services that can produce the superattributes of the given attribute. Thus, the number of the lookup operations increases rapidly. For example, in data set #4 that contains nearly one million *subattribute-of* relations, there might be thousands of superattributes (i.e., subtypes) for a given attribute. Moreover, in data set #4, the number of types (the key of Map_D and Map_O) is much larger than the number of services (the key of Map_Δ), which makes the lookup operation on Map_Δ much faster than that on Map_D and Map_O. As a result, t_e is significantly increased, so is the effectiveness factor λ. WS-Challenge 2007 also supports inheritances in XML schema, and shares the data sets with WS-Challenge 2006, so this improved solution is valid. With this improved solution as well as many other improvements on program code and architecture, we finally won the championship of performance in the competition of WS-Challenge 2007.

One interesting phenomenon in the experiment results is that t_e of request No. 11 of data set #4 is notably larger than 298 ms, the preprocessing time cost of the data set. This occurs because dependencies are discarded across iterations in the baseline solution, so some dependencies may be constructed more than one time. This suggests that the sharing of dependencies is important even to the processing of a single composition request.

Also note that this improved solution is general to data-driven service composition algorithms. For example, in WS-Challenge 2008, although the problem definition is shifted from constructing service chains to constructing service graphs, this improved solution is still applicable. And it is obvious that the more complicated service data correlations are, the more improvements on performance will be made.

7 Related Work

Service data correlations are usually expressed in an implicit way in related works. The most straightforward way is to deduce data correlations by matchmaking on data models used for service I/O. In addition to the SDG model proposed by Liang and Su [2], [1], Oh et al. [11] adopted STRIPS, the input language of an automated solver (also called STRIPS) for AI-planning, to model a kind of data-driven service composition problem. The input/output attributes of a service operation in the STRIPS model are modeled as the pre/postconditions of a state in which the corresponding service operation can be invoked. A polynomial-time algorithm, WSPR, is proposed to solve the STRIPS model. In SWORD [9], the I/O of a service is divided into "Data" and "Condition" parts, and a rule-based expert system is adopted to create desired service composition automatically. The WS-Challenge participants proposed various methods to solve the challenge problem by traversing the data correlations implied by the data sets [13], [14], [15], [16], [17], [18].

As data models are usually not strict on data semantics, people of the semantic web services [19] community suggest a variety of methods to model service I/O with ontology. The METEOR-S [20] project proposed two annotation frameworks for WSDL, WSDL-S [21], [22] and SAWSDL [23], [24]. Unlike these two annotation frameworks, OWL-S [25] and WSMO/WSML [26], [27] try to establish standards that build service descriptions directly on ontology. Given ontology annotations and descriptions, we can deduce data correlations by reasoning on ontology of service I/O, which will be more powerful than matchmaking on data models, as ontology is designed to be reasonable.

However, no matter data correlations are implied by ontology or by data models, their expressive power is limited due to the restrictions presented in Section 5.3. Thus, the key contribution of this work is that we figured out this fundamental problem, and proposed a method to specify service data correlations by using explicit declarations, and that is the basis of the two models, SDG+ and SDL, proposed in this paper.

Also we noticed that there are some works that provide the mechanism to specify data correlations explicitly, for example, the XLink [28] specification and the Active XML [29], [30] project. However, both of them focus on data correlations among documents, which is significantly different from data correlations among service operations.

8 Conclusion

Service data correlation is essentially a possible dataflow between two service operations, which is an important kind of information to service composition and related tasks such as service discovery and selection. However, service data correlations are usually expressed implicitly through service definitions and data models, or informally described in technical documents written in natural languages. In such ways, the power of data correlation cannot be fully recognized and leveraged. In this paper, we proposed two models, SDL and SDG+, to describe data correlations among service operations. SDL is a general purpose model dedicated to service data correlation modeling, while SDG+, a combination of SDL and SDG, is a domain model, which particularly focuses on data-driven service composition.

Despite the details of these two models, we say that the key idea underlying them is explicit declaration of data correlations. We have shown in the paper that SDL is highly comparable to hyperlink; further, to some extent, hyperlink is a kind of explicit data correlation declaration, which is usually expressed in a loose manner and is supposed to be *complete* to its target. From this viewpoint, we believe that explicit declaration is a general purpose concept for data correlation modeling. With explicit declaration, people may create dedicated models to describe complex data correlations among various entities, such as webpages, web services, distributed objects, which are the core components of SOC applications.

Acknowledgments

This work is supported by the National Basic Research Program of China (973) under grant No. 2007CB310803 and the China National High-Tech Project (863) under grants No. 2007AA010306 and No. 2009AA01Z120. This work is also inspired by the WS-Challenge competition.

REFERENCES

[1] Q. Liang, L.N. Chakarapani, S.Y.W. Su, R.N. Chikkamagalur, and H. Lam, "A Semi-Automatic Approach to Composite Web Services Discovery, Description and Invocation," *Int'l J. Web Services Research,* vol. 1, no. 4, pp. 64-89, 2004.

[2] Q.A. Liang and S.Y.W. Su, "AND/OR Graph and Search Algorithm for Discovering Composite Web Services," *Int'l J. Web Services Research,* vol. 2, no. 4, pp. 48-67, 2005.

[3] E. Rahm and P.A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," *The VLDB J.,* vol. 10, no. 4, pp. 334-350, 2001.

[4] H.H. Do, S. Melnik, and E. Rahm, "Comparison of Schema Matching Evaluations," *Proc. NODe Web and Database-Related Workshops Web, Web-Services, and Database Systems,* pp. 221-237, 2003.

[5] S. Tyagi, "Patterns and Strategies for Building Document-Based Web Services," http://java.sun.com/developer/technicalArticles/xml/jaxrpcpatterns, Sept. 2004.

[6] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," http://www.w3.org/TR/wsdl, Mar. 2001.

[7] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, "Automatic Composition of E-Services that Export Their Behavior," *Proc. First Int'l Conf. Service-Oriented Computing (ICSOC '03),* pp. 43-58, 2003.

[8] D. Berardi, D. Calvanese, G.D. Giacomo, M. Lenzerini, and M. Mecella, "ESC: A Tool For Automatic Composition of E-Services Based on Logics of Programs," *Proc. Fifth VLDB Int'l Workshop Technologies for E-Services (VLDB-TES '04),* pp. 80-94, 2005.

[9] S.R. Ponnekanti and A. Fox, "SWORD: A Developer Toolkit for Web Service Composition," *Proc. 11th Int'l Conf. World Wide Web (WWW '02),* http://www2002.org/CDROM/alternate/786, 2002.

[10] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid, "Composing Web Services on the Semantic Web," *The VLDB J.,* vol. 12, no. 4, pp. 333-351, 2003.

[11] S.-C. Oh, D. Lee, and S.R.T. Kumara, "Web Service Planner (WSPR): An Effective and Scalable Web Service Composition Algorithm," *Int'l J. Web Services Research,* vol. 4, no. 1, pp. 1-23, Jan.-Mar. 2007.

[12] B. Xu, T. Li, Z.F. Gu, and G. Wu, "SWSDS: Quick Web Service Discovery and Composition in SEWSIP," *Proc. Eighth IEEE Int'l Conf. E-Commerce Technology/Third IEEE Int'l Conf. Enterprise Computing, E-Commerce, and E-Services (CEC/EEE '06),* pp. 71-71, 2006.

[13] S. Huang, X.L. Wang, and A.Y. Zhou, "Efficient Web Service Composition Based on Syntactical Matching," *Proc. IEEE Int'l Conf. E-Technology, E-Commerce, and E-Services (EEE '05),* pp. 782-783, 2005.

[14] M. Aiello, C. Platzer, F. Rosenberg, H. Tran, M. Vasko, and S. Dustdar, "Web Service Indexing for Efficient Retrieval and Composition," *Proc. Eighth IEEE Int'l Conf. E-Commerce Technology/Third IEEE Int'l Conf. Enterprise Computing, E-Commerce, and E-Services (CEC/EEE '06),* pp. 63-63, 2006.

[15] S.-C. Oh, B.-W. On, E.J. Larson, and D. Lee, "Bf*: Web Services Discovery and Composition as Graph Search Problem," *Proc. IEEE Int'l Conf. E-Technology, E-Commerce, and E-Services (EEE '05),* pp. 784-786, 2005.

[16] S.-C. Oh, H. Kil, D. Lee, and S.R.T. Kumara, "Algorithms for Web Services Discovery and Composition Based on Syntactic and Semantic Service Descriptions," *Proc. Eighth IEEE Int'l Conf. E-Commerce Technology/Third IEEE Int'l Conf. Enterprise Computing, E-Commerce, and E-Services (CEC/EEE '06),* p. 66, 2006.

[17] M.A. Makhzan and K.-J. Lin, "Solutions to a Complete Web Service Discovery and Composition," *Proc. Eighth IEEE Int'l Conf. E-Commerce Technology/Third IEEE Int'l Conf. Enterprise Computing, E-Commerce, and E-Services (CEC/EEE '06),* pp. 73-73, 2006.

[18] Y. Zhang, T. Yu, K. Raman, and K.-J. Lin, "Strategies for Efficient Syntactical and Semantic Web Services," *Proc. Eighth IEEE Int'l Conf. E-Commerce Technology/Third IEEE Int'l Conf. Enterprise Computing, E-Commerce, and E-Services (CEC/EEE '06),* p. 72, 2006.

[19] S.A. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems,* vol. 16, no. 2, pp. 46-53, Mar./Apr. 2001.

[20] A.A. Patil, S.A. Oundhakar, A.P. Sheth, and K. Verma, "METEOR-S Web Service Annotation Framework," *Proc. 13th Int'l Conf. World Wide Web (WWW '04),* pp. 553-562, 2004.

[21] K. Sivashanmugam, K. Verma, A.P. Sheth, and J.A. Miller, "Adding Semantics to Web Services Standards," *Proc. First Int'l Conf. Web Services (ICWS '03),* L.J. Zhang, ed., pp. 395-401, 2003.

[22] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.-T. Schmidt, A. Sheth, and K. Verma, "Web Service Semantics—WSDL-S," http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.pdf, Apr. 2005.

[23] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," *IEEE Internet Computing,* vol. 11, no. 6, pp. 60-67, Nov. 2007.

[24] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema," W3C Working Draft, http://www.w3.org/TR/2007/WD-sawsdl-20070410, Apr. 2007.

[25] D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D.L. Mcguinness, E. Sirin, and N. Srinivasan, "Bringing Semantics to Web Services with OWL-S," *World Wide Web,* vol. 10, no. 3, pp. 243-277, 2007.

[26] J. de Bruijn, D. Fensel, U. Keller, and R. Lara, "Using the Web Service Modeling Ontology to Enable Semantic e-Business," *Comm. ACM,* vol. 48, no. 12, pp. 43-47, 2005.

[27] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler, "WSMX—A Semantic Service-Oriented Architecture," *Proc. Third IEEE Int'l Conf. Web Services (ICWS '05),* pp. 321-328, 2005.

[28] S. DeRose, E. Maler, and D. Orchard, "XML Linking Language (XLink) Version 1.0," W3C Recommendation, http://www.w3.org/TR/xlink, June 2001.

[29] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber, "Active XML: Peer-to-Peer Data and Web Services Integration," *Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02),* pp. 1087-1090, 2002.

[30] S. Abiteboul, O. Benjelloun, and T. Milo, "The Active XML Project: An Overview," *The VLDB J.,* vol. 17, no. 5, pp. 1019-1040, 2008.

Zhifeng Gu received the PhD degree in computer science from Tsinghua University, P.R. China, in 2009, and is currently working at Baosight, Inc., as a software engineer. He participated in the Web Service Challenge at the IEEE CEC 2006 and 2007 conferences and won the championships of both. His research interests include service discovery and composition, semantic web services, SOA/SOC. Recently, he has become especially interested in enterprise computing technologies including BRMS and cloud computing. He is a student member of the IEEE.

Bin Xu received the PhD degree in computer science in 2006 from Tsinghua University, P.R. China, where he is currently working as an associate professor in the Department of Computer Science and Technology. His research interests include web service composition, semantic web service, and sensor networking. He won the championships of the Web Service Challenge at the IEEE CEC 2006, 2007, and 2008 conferences. He is a member of the IEEE and the ACM.

Juanzi Li received the PhD degree in computer science in 2000 from Tsinghua University, P.R. China, where she is currently working as a professor in the Department of Computer Science and Technology. Her research interests include semantic web and semantic web services, and text and social network mining. She is a member of the IEEE and a professional member of the ACM.