Semantic Web service composition through a Matchmaking of domain

Freddy Lécué, Alain Léger
France Telecom R&D, 4 rue du clos courtel,
F-35512 Cesson Sévigné, France
{(freddy.lecue, alain.leger)@orange-ft.com}

Abstract

The automated composition of Web services is one of the most promising ideas and at the same time one of the most challenging research area for the taking off of service-oriented applications. Services which are composed automatically need to share a variety of important characteristics such as a common understanding of their semantics capabilities. It is widely recognized that one of the key elements for the automated composition of Web services is semantics: i.e. unambiguous descriptions of Web services capabilities and web service processes. For example proposed standard languages like OWL-S, WSMO, WSDL-S or FLOWS can provide the ability to reason about Web services, and to automate web services tasks, like discovery, selection and composition. However Web services described at capability level need a formal context to perform the automated composition of Web services. In this paper an architecture and a semantic model for Web service composition are presented.

1. Introduction

An important vision of service oriented computing is to enable semantic, dynamic service binding, i.e. it should become possible to automatically choose and invoke service providers at runtime.

Service Oriented Architecture (SOA)-based Web Service provides a suitable technical foundation for loosely coupled and reusable software components. However there is still some work to be done to appropriately support dynamic and automated tasks such as discovery, selection and composition. First of all the main requirement is the ability to describe capabilities (i.e. Input, Output, PreConditions and Effects IOPEs), process models (i.e. providing a description of Web service activities, interaction protocol, exchanged messages) and grounding specification (i.e. abstraction of information exchanged mapped onto messages) of Web service. Such a semantic value-added may be covered by means of semantic Web service [35]. A semantic Web service is seen as a web service whose description is in a language that has well-defined semantics. Thus using of semantics enables inference about the Web service requirements and effects, which in turn facilitates automatic discovery, selection, composition and reasoning.

Towards automatically advising new value-added Web services we focus on Web service composition which will be expressed on three main levels. The first component is Web service Discovery or the process of locating a set of relevant Web services. The second unit is Functional level composition [15, 20, 27] which considers Web services as “atomic” components described in terms of their inputs, outputs, preconditions, and effects, which can be executed in a simple request-response step. Finally Process level composition [8, 9, 25, 29] automatically generates an executable composite Web service.

The rest of the paper focuses on functional level composition. In section 2 we discuss the architecture of the web service composer. Section 3 presents a motivating example through an e-healthcare scenario. In section 4 we introduce and explain our solution proposal for the functional level composition problem. Section 5 introduces the causal link matrix and the composition process. We briefly comment on related work in section 6 and finally in section 7 we draw some conclusions and talk about possible future directions.

2 TLBA: A Three Levels based Architecture

Three main issues are involved in the Web service composition. First of all Web service discovery aims at reaching the user goal. Such an issue not only include the discovery process but also the Web service selection step. The second challenge is to propose a workflow (or a simplified form of a workflow: plan) which describes how Web services interact and how the functionality they offer could be orchestrated and monitored to provide realistic solutions. The last point refers to interaction, conversation and choreography management of Web services. The execution process is out of scope of this paper, indeed such a process is supposed to be
achieved once a Web service composition is proposed and valid. That is why a Three Levels Based Architecture (Figure 1) is suggested to tackle the Web service composition challenge. As previously explained, the TLBA is divided into three modules: the Web services discover, the functional level composer and the process level composer [30]. The main components are described with more details in the subsections below.

2.1 Web service discovery

The Web services discover is able to locate suitable Web services for a given request. Web Service discovery often referred to service matchmaking given a query specifying inputs and outputs of the desired Web service. The problem is to discover a list of “best matched” advertisement services in a service registry. The discovery process is far from easy going because the UDDI registries standard have not yet explicit representation of the whole semantic description of Web services. From a semantic view, it is extremely difficult to find relevant Web services, given a desired capability. However some research effort proposes model to automate discovery process such as [6, 10, 19, 26]. Even if such a module is in the TLB architecture, this unit is out of scope of the paper.

2.2 Functional and Process composition

In this paper two different and complementary approaches [1, 29] are considered to solve a Web service composition problem. The first method is capability-oriented (or data flow driven) [24, 35, 27, 33, 14, 20] whereas the second solution is process-oriented (or control flow driven) [8, 9, 25, 29]:

- the functional level composition selects a set of services that if combined in a suitable way, are able to match a given objective. Thus functional level composition is usually combined with service discovery techniques, that are expressed to find suitable service instances to be chained [15]. This composition model outlines the Web service inputs, outputs, preconditions, and effects (IOPEs) requirements. For instance the OWL-S[3] functional level is presented by the service profile, whereas this level is presented by the capability model in a WSMO[17] specification.
- Another open problem for semantic Web services is composition at process level, i.e. the problem of generating automatically composed Web services that can be directly executed to interact with Web services and achieve a composition goal. Process level composition defines an interaction pattern with selected services, so that an executable implementation of the composition is obtained. For instance a Web service is defined by the process level in OWL-S, whereas the same entity is described as an activity flow or an interaction pattern in WSMO known as the “interface model”. The key elements of such a composition are objects manipulated by Web services (actions) i.e. typed messages with complex descriptions (described according to STS[29], π-calculus, or Process Algebra [5]).

In contrast to the data flow driven approach where the partial order of execution tasks is specified implicitly, execution order is specified explicitly with control constructs (modelled by STS, π-calculus, or Process Algebra) in a control flow driven approach. The two previous composition models may be an interesting tradeoff for the Web service composition part in the TLBA.

2.3. Web service composition approach

According to the Figure 1, the $D_4AC^1$ algorithm is a sketch of a Web service composition algorithm wherein the knowledge base $KB$ refers to ABox elements, i.e. instantiated concepts described in a terminology $T$. The goal $β$ refer to concepts in the same terminology $T$. The latter algorithm is composed of two main parts: the functional level (III) and process level composition (IV). However Web service discovery and selection are necessary steps to deal with Web service composition. In the following functional level composition is studied in more details. Moreover the Problems Mapping (section 4) and Causal link Matrix issues (section 5) will be described in more details.

3. A motivating example

3.1. An e-healthCare scenario

Let us illustrate a Web service composition in a Machine to Machine system with teledmedical collaborations of medical devices. The reduction of additional consultation, examination, medical check up fees and the improving of the

---

$^1$Acronym for Decomposition for Automatic Composition.
patient follow-up are the main motivations of this scenario. A complete and whole clinical observation in hospital is no longer a realistic issue for cost reasons, especially for the elderly. A solution of the previous problem consists of implementing a composite and value-added Web service that can automate the patient follow-up by a reliable Web service interoperation, hence a long distance follow-up. With the aim to automate an effective and reliable assistance in case of clinical problems appear, a TLB architecture is submitted. Thus the Web services discover module discovers and selects all relevant Web services in order to automate composition of Web services. Indeed the Web services discover module selects the best suitable Web services set according to an end user request. The functional level composer does not deal with the discovery process.

For a SOA view, Web services are independently located on different machine. Such a simple but realistic scenario highlights challenges of functional level composition.

Algorithm 1: D_AC Algorithm.


Result: The best Plan solution P.

begin

(I) Request analysis and goal discovery (β);
(II) Web service discovery with BCov(T, Q);
(III) Goal analysis and decomposition;
(IV) Functional level composition;
   (i) Problems mapping (Composition, Matching);
   (ii) Causal link matrix discovery;
   (iii) Find a complete, correct, consistent and optimal plan P for a planning problem < S_ws, KB, β >;
(V) Process level composition and P adaptability;

end

Consider the above scenario with six different Web services (Table 1): S_a returns the blood pressure (BP) of a patient given his PatientID (PID) and DeviceAddress (Add); S_b and S_d return respectively the supervisor (Person) and a physician of an organisation; S_c returns a level of warning (WL) given a blood pressure; S_d returns the Emergency department (Emerg. Dpt) given a level of Warning; S_e returns the Organization (Org.) given a level of Warning (WL). Each Web service are supposed to the most relevant services from a discovery point of view [6]. All parameters (i.e. Input and Output) referred to concept in a terminology T (Figure 2).

The main purpose of patient follow-up is to determine whether or not the treatment is effective enough. In case of emergency, human (physician and professional) will take place in the medical process.

<table>
<thead>
<tr>
<th>Web Services</th>
<th>S_a</th>
<th>S_b</th>
<th>S_c</th>
<th>S_d</th>
<th>S_e</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>PID, Add</td>
<td>Org.</td>
<td>BP</td>
<td>WL</td>
<td>WL</td>
</tr>
<tr>
<td>Precondition</td>
<td>Pre_S_a</td>
<td>Pre_S_b</td>
<td>Pre_S_c</td>
<td>Pre_S_d</td>
<td>Pre_S_e</td>
</tr>
<tr>
<td>PostCondition</td>
<td>Post_S_a</td>
<td>Post_S_b</td>
<td>Post_S_c</td>
<td>Post_S_d</td>
<td>Post_S_e</td>
</tr>
</tbody>
</table>

Table 1. Semantic Web services of S_ws and their capabilities.

4. Formal context and representation

A formal model is introduced in order to formalize Web service composition at functional level. The employed formalism for Web service description may be easily mapped to the OWL-S or WSMO specification.

4.1 Web service as a specific function

According to the WSDL standard definition [11], each Web service may contain definitions of various operations. In the following, Web Service operations and Web Services will refer to same entities. Thus each Web service refers to a single operation, and each operation stands for a single Web service. A Web Service operation or a Web Service is described by its name, the knowledge of its parameters (input and conditional output parameters), and by its preconditions and effects (i.e. post-conditions) according to the OWL-S and WSMO specification. A simple and effective formal definition of a Web service i.e. W_w(Inputs, Outputs, Preconditions, Effects), is proposed to Web service composition at functional level.

The previous definition introduces a Web service as an entity able to produce one or more concrete result(s) outputs according to necessary inputs. Preconditions specify things of the world that must be true in order to execute a Web service. Conditional effects characterize physical side-effects, execution of a Web service has on the world. Input parameters are used to pass information to the task when it
is started whereas output parameters are filled with information returned from the task once it is performed. Contrary to preconditions and conditional effects which are described in a first order logic (necessary for reasoning about facts), inputs and conditional outputs are described in Description Logics [4]. Thus each Web service parameter refers to a concept in a specified ontology T. A generalization of the Web service definition is:

\[ s_x : \{ \text{Input}^{n}_x \rightarrow \text{Output}^{1}_x \ (i^1_x, \ldots, i^{n}_x) \rightarrow p_x \ (o^1_x, \ldots, o^{m}_x) \} \]  

(1)

In such a representation, a Web service \( s_x \) has \( \alpha \) input parameters \( i^1_x, \ldots, i^{n}_x \) and \( \gamma \) conditional output parameters \( o^1_x, \ldots, o^{m}_x \). Furthermore all parameters are high-level concepts defined in a specific ontology \( T \). Finally the relationship \( \rightarrow p_x \) highlights the preconditions and effects features described with first order logic axioms. The suggested model aims at not only handling input and output parameters but also controlling the state of the world [31] of Web services.

**Example 1.** According to the formal model (1), Web services \( S_c \) and \( S_d \) are respectively modeled by \( S_c \) (In BloodPressure, Out Warning Level, Pre\( S_c \), Post\( S_c \)) and \( S_d \) (In Warning Level, Out Emergency Dpt, Pre\( S_d \), Post\( S_d \)). One precondition for \( S_c \) may be \( \text{Valid} \) (BloodPressure) (Table 1).

### 4.2 Web service Composition formalism

Once the Web service model introduced, a simple but realistic composition of two Web services \( s_y \) and \( s_x \) is investigated as an application \( s_y \circ s_x \) defined by:

\[ s_y \circ s_x : \{ \text{Input}^{n}_x \rightarrow \text{Output}^{1}_x \ (i^1_x, \ldots, i^{n}_x) \rightarrow p_x \ (o^1_x, \ldots, o^{m}_x) \} \]  

(2)

with \( s_y \) and \( s_x \) are defined by (1), respectively with \( (\alpha, \gamma) \) equals to \((m, l)\) and \((n, k)\). Moreover the output parameters of \( s_x \) and the input parameters of \( s_y \) match in a semantic context, e.g. by the subsumption relationship. In other words, Web service composition consists on a matchmaking of Web services domains: the output domain \( \text{Output}^{k}_x \) of \( s_x \) and the input domain \( \text{Input}^{m}_y \) of \( s_y \). A Web service composition of \( s_y \circ s_x \) is possible only if effects of \( s_x \) satisfy all necessary preconditions of \( s_y \).

Even if only one simple composition of two Web services is presented, this model could be easily extended to more complex composition types. However for space reasons we present and illustrate only the composition problem of two Web services such as effects of \( s_x \) satisfy preconditions of \( s_y \).

Example 2. The Web services \( S_c \) and \( S_d \) may be composed as \( S_y = S_d \circ S_c \) such that \( S_d \circ S_c \) (In BloodPressure, Out Emergency Dpt, Pre\( S_d \), Post\( S_d \)). Moreover the output parameter of \( S_c \) (i.e. Warning Level) and the input parameter of \( S_d \) (i.e. Warning Level) match with the equivalence relationship.

### 4.3 Functional level composition

According to the previous considerations, functional level composition is considered as a problem of Domains Matchmaking. A solution consists of finding the best “match” between two functional domains. The first entity refers to the \( \text{Output}^{1}_x \) range whereas the second entity refers to the \( \text{Input}^{m}_y \) domain. So a Web service composition problem is rewritten as a Matchmaking problem of (Range, Domain). Therefore solving a \( s_y \circ s_x \) composition problem is equivalent to find a relevant Matchmaking (and also “Matching functions”) between \( k \) output parameters \( (o^1_x, \ldots, o^{m}_x) \) of a Web service \( s_x \) and \( m \) input parameters \( (i^1_y, \ldots, i^{n}_y) \) of a Web service \( s_y \).

![Figure 3. Problems Mapping.](image)

Solving the previous problem is related to find a Web service \( s_x \) (or more Web services in case \( k \neq m \)) such that its output parameters have a “conditional matching”\(^3\) with the input parameters of another Web service \( s_y \). In other words, Web service composition \( s_y \circ s_x \) is similar to discover a Web service \( s_y \) such that the domains of \( \text{Inverse}_S(s_y) \) (i.e. \( I_S(s_y) \)) and \( s_x \) semantically match (Figure 3). The \( I_S \) application is defined by \( I_S \):

\[ I_S : \{ \text{Input}^{n} \times \text{Output}^{m} \rightarrow \text{Output}^{\text{Card}KB} \times \text{Input}^{n} \ (\ ((i^1_x, \ldots, i^{n}_x), (o^1_y, \ldots, o^{m}_y)) \rightarrow c_x ((KB_x), (i^1_x, \ldots, i^{n}_x)) \} \]  

(3)

with \( KB_x \) refers to a knowledge base \( KB \) such that \( KB_x \equiv (KB_1, \ldots, KB_{\text{Card}KB}) \). \( KB_{\text{Card}KB} \) allows the initialisation of the input domain of the inverse Web service of \( s_y \). Thus the input parameters and preconditions of a Web service \( s_y \) become the output parameters and effects of

\(^3\)The easy case deals with \( k = m \).

\(^3\)The “conditional matching” refers to a Matching application.
a Web service $I_S(s_y)$ whereas the $s_y$ outputs and effects are not mapped to $I_S(s_y)$ parameters. Each Web service transformed of $I_S$ does not have any preconditions. This choice is justified because the $s_y$ output parameters and effects are supposed to be not relevant for a composition $s_y \circ s_x$. Thus $I_S$ application translates a functional composition problem into a Web services matching problem. Composing two Web services $s_y$ and $s_x$ is equivalent to find an appropriate Matching between $I_S(s_y)$ and $s_x$. Figure 3 presents the introduced problems mapping.

**Example 3.** According to example 2, the Web services $S_c$ and $S_d$ may be composed as $S_d \circ S_c$ if there exists a semantic matching between the Web service $S_c$ and the abstract Web service $I(S_d)$, i.e. between the Web service $S_c$ and an abstract Web service. The latter abstract Web service have instantiated concepts of KB as input parameters. Their output parameters are the input parameters of $S_d$. $I(S_d)$ is defined by $I(S_d)(\text{In } KB_x, \text{Out } WL, \emptyset, \text{Pre}_S_d)$.

The Reference Abstract Service for Composition $S_{RASC}$ is introduced to explain in more details the new problem.

**Definition 1.** A Reference Abstract Service for Composition $S_{RASC}$ of $s$ is defined as an abstract Web service, transformed of $s$ by the $I_S$ application. The $S_{RASC}(s)$ output parameters are exactly the input parameters of $s$, and the input parameters of $S_{RASC}$ are initialized with the KB elements (by default).

In the same way $S_{HWS}^{i_1}(s)$ is defined as a transformed of $s$ by the $I_S$ application which returns a specialized Web service with constraints on output parameters (actually only $i_1$ as output parameter). In definition 1 and 2, the Input (Output) and Precondition (Effect) meaning may exchanged without loss of generality.

**Example 4.** Regarding the running example, $S_{RASC}(S_d)$ is functionally described by $S_{RASC}(S_d)(\text{In } KB_x, \text{Out } WL, \emptyset, \text{Pre}_S_d)$.

**Definition 2.** An Helper Web Service $S_{HWS}^{i_1}(s)$ is defined as an abstract Web service, transformed of $s$ by the $I_S$ application with only one output parameter $i_1$.

The output parameters of $S_{HWS}^{i_1}(s)$ are reduced to the $\{i_1\}$ singleton, and input parameters of $S_{HWS}^{i_1}(s)$ are instantiated with KB. Thus a Web service, transformed of $s$ by $S_{RASC}$ application is a specialized version (with more output parameters) of a Web service, transformed of $s$ by $S_{HWS}$ application.

**Example 5.** $S_{HWS}^{\text{BloodPressure}}(S_c)$ is functionally described by $S_{HWS}^{\text{BloodPressure}}(S_c)(\text{In } KB_x, \text{Out BloodPressure}, \emptyset, \text{Pre}_S_c)$.

A new problem of Web service Matching at functional level (i.e. Input and Output Parameters) is introduced to solve a Web service composition. Thus it is necessary to propose distinct and various Matching functions in order to be able to compute different Matching between Web services (i.e. kind of Web services composition solution). The matching functions are presented in Table 2.

### 4.4. Matching functions and composition

It is necessary to compose Web services using appropriate Matching properties with the purpose of obtaining an automated and suitable Web service composition. Composing $s_y$ and $s_x$ according to $s_y \circ s_x$ will be similar to find the best matching type between $\text{Inverse}_S(s_y)$ and $s_x$. This matching evaluation will be computed according to Table 2.

<table>
<thead>
<tr>
<th>Match(S, Q)</th>
<th>Definition</th>
<th>Sim$_T$(S,Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{\text{in}}^{\text{out}}(S, Q)$</td>
<td>$(\text{Q}<em>{\text{in}} \rightarrow \text{S}</em>{\text{in}}) \land (\text{Q}<em>{\text{out}} \rightarrow \text{S}</em>{\text{out}})$</td>
<td>1</td>
</tr>
<tr>
<td>$2. M_{\text{exact}}^{\text{out}}(S, Q)$</td>
<td>$(\text{Q}<em>{\text{in}} \rightarrow \text{S}</em>{\text{in}}) \land (\text{Q}<em>{\text{out}} \rightarrow \text{S}</em>{\text{out}})$</td>
<td>0.6</td>
</tr>
<tr>
<td>$3. M_{\text{exact}}^{\text{out}}(S, Q)$</td>
<td>$(\text{Q}<em>{\text{in}} \rightarrow \text{S}</em>{\text{in}}) \land (\text{Q}<em>{\text{out}} \rightarrow \text{S}</em>{\text{out}})$</td>
<td>0.6</td>
</tr>
<tr>
<td>$4. M_{\text{plug-in}}(S, Q)$</td>
<td>$(\text{Q}<em>{\text{in}} \rightarrow \text{S}</em>{\text{in}}) \land (\text{Q}<em>{\text{out}} \rightarrow \text{S}</em>{\text{out}})$</td>
<td>0.4</td>
</tr>
<tr>
<td>$5. M_{\text{plug-in}}^{\text{out}}(S, Q)$</td>
<td>$(\text{Q}<em>{\text{out}} \rightarrow \text{S}</em>{\text{out}})$</td>
<td>0.2</td>
</tr>
</tbody>
</table>

**Table 2. Matching functions summary.**

The Matching functions 1, 4 and 5 were previously studied in the community research of software components [36] (i.e. $C \leftrightarrow S$, with $C$ refers a software component). However two more relevant Matching functions $M_{\text{exact}}^{\text{out}}$ and $M_{\text{exact}}^{\text{out}}$ are introduced. These two new functions add substantial power to obtain a finer control on the discovery and selection process for Web service composition. Indeed Web service composition has locally changed into a Web service discovery problem through a problem of Web service matchmaking.

**Example 6.** Suppose the Web service $S_a$ from the motivating example and another Web service $S_{\text{tmp}}(\text{In PatientID, Out BloodPressure, Pre}_{S_{\text{tmp}}}, \text{Post}_{S_{\text{tmp}}})$. According to Table 2, $M(S_{\text{tmp}}, S_a) = M_{\text{exact}}^{\text{out}}$. So $\text{Sim}_T(S_{\text{tmp}}, S_a) = 0.6$

In the same way, $M(S_c, S_d) = M_{\text{exact}}^{\text{out}}$ because the concept EmergencyDpt is semantically subsumed by the concept Organization according to an ontology domain $T$. So $\text{Sim}_T(S_c, S_d) = 0.6$.

Even if the Matching function detail is out of scope of this paper [36], a partial order is proposed to valued semantic quality of Matching (Theorem 1).

**Theorem 1.** Let the five functions in Table 2, one have the following relations on the Matching functions with $R$ refers to a binary relation (i.e. logical implication $\rightarrow$):
5. Causal link matrix

According to the previous section, theorem and property, we are able to find a simple Web service composition such as \( s_y \circ s_x \). However this solution is far from expressive enough to solve complex composition problems. That is why causal link matrices (i.e. Matching matrix) as a formal model is introduced. The key contribution of a causal link matrix is to obtain a formal and robust basis to control and compose a finite set of relevant Web services. Thus causal link matrices contribute to the automated process of Web service composition by classifying Web services according to a formal link called “causal link”. A causal link is related to a logical dependency among input and output parameters of different Web services.

This section focuses on part III)ii of the \( D_4AC \) algorithm and more specifically on the definition of causal link matrices.

### 5.1. Motivation

Algorithms for Web service composition have to not only find feasible plans with relevant Web services, but also find the optimal plan according to an optimization criteria. The latter criteria will be viewed as a quality of semantic connection between Web services (Input and output parameters relation). Indeed the semantic connection between Web services is considered as essential to form new value-added Web services. The formal model (i.e. the Causal link matrices CLMs) aims at storing all those connections in order to find the best Web service composition. The CLM pre-computes all semantic links (Table 2) between Web services as an Output-Input matching because a Web service composition is mainly made up of semantic connections. Indeed a solution of a Web service composition have to design and define a plan of Web services wherein all Web services are semantically well ordered and well linked. The latter links are computed and stored in CLMs.

The idea behind the CLM is a formal model to store Web services in an adequate and semantic context for functional level composition of Web services. The CLM aims at proposing a composition model for a finite set of Web services. The latter Web services are supposed to be relevant according to a discovery criteria [7, 35]. In such a case, the CLM pre-computes and defines all the possible semantic matchings between Web services to improve the performance of Web service composition, and also to make Web service composition easier. Moreover CLMs allow us to consider a simpler composition problem, i.e. the causal link composition. Thus the Web service composition is mapped to a causal link composition wherein causal links inform about semantic connections between Web service. A Web service composition solution is mainly oriented by the CLM of the domain.

### 5.2. Definitions and Algorithm

A causal link matrix contains all enabled, legal and valid transitions for a composition goal because causal links help to detect inconsistencies of semantic link between Web services. Indeed all valid causal links between Web services are explicitly represented with a value pre-computed by the \( \text{Sim}_F \) function (Table 2). The latter value is based on the semantic quality of matching between two Web services. The Causal link matrix aims at storing all those valid causal link in an appropriate way. The more valid causal links there are, the better it is for a functional composition problem.

**Definition 3.** (Causal link matrix CLM)

The set of \( p \times q \) CLMs\(^4\) is defined as \( M_{p,q}(P((S_{WS} \cup \)\)

\(\text{\#}(S)\) refers to the set of parts of a set \( S \). \( \text{\#}(S) \) refers to the Cardinality of \( S \).
Warning Level

<table>
<thead>
<tr>
<th>i(j ) index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i, label)</td>
<td>Address (Add)</td>
<td>BloodPressure (BP)</td>
<td>Organization</td>
<td>Patient</td>
</tr>
<tr>
<td>(c, label)</td>
<td>Address (Add)</td>
<td>BloodPressure (BP)</td>
<td>Organization</td>
<td>Patient</td>
</tr>
</tbody>
</table>

Table 3. Labels of the rows \(r_i\) and columns \(c_j\) of the \(5 \times 6\) matrix \(\mathcal{M}\).

\(T \times [0, 1]\)). Columns \(c_{j \in \{1,...,q\}}\) are labelled by \((\text{Input}(S_{W,s}) \cup \beta) \subseteq T\), the inputs of parameters of services in \(S_{W,s}\) and/or the concepts described by the goal set \(\beta \subseteq T\). Rows \(r_{i \in \{1,...,p\}}\) are labelled by \((\text{Input}(S_{W,s}))\), the inputs of parameters of services in \(S_{W,s}\). Each entry \(m_{i,j}\) of a CLM \(\mathcal{M}\) is defined as a set of pairs \((s_y, \text{score}) \in (S_{W,s} \cup T) \times [0, 1]\).

\[
(s_y, \text{score}) = \begin{cases} 
(s_y, \text{Sim}_T(s\text{In } r_i, \text{Out } c_j), s_y)_{s_y \in S_{W,s}} \\
(s_y, 1)_{s_y \in T}
\end{cases}
\]

with \(r_i \in T \cap \text{In}(s_y) \subseteq \text{Input}(S_{W,s})\) is the label of the \(i^{th}\) row,

with \(c_j \in T \cap (\text{Input}(S_{W,s}) \cup \beta)\) is the label of the \(j^{th}\) column.

Out \((s_y)\) is the set of output parameters of the Web services \(s_y\) whereas \(\text{In}(s_y)\) is its set of input parameters. \(\beta\) contains the set of goals, described as concepts in a terminology \(T\). Those concepts have to be reached. The variable \(\text{score}\) refers to the degree of match \(\text{Sim}_T(s_y, s\text{In } r_i, \text{Out } c_j)\) between a Web service \(s_y \in S_{W,s}\) and an abstract Web service \(s\) with \(r_i\) as an input parameter, \(c_j\) as an output parameter in case \(s_y \in S_{W,s}\). In the alternative case \(s_y \in T\), the value \(\text{score}\) is 1. A CLM pre-computes the semantic similarities between all output and input parameters of a closed set of Web services. All entries defined in \(\mathcal{P}((S_{W,s} \cup T) \times [0, 1])\) are positive and valid causal links.

A CLM is seen as a matrix with entries in \(\mathcal{P}((S_{W,s} \cup T) \times [0, 1])\). Thus each entry of a CLM refers to a set of pairs \((s_y, \text{score})\) such that the score refers to a semantic similarity between two Web services. All semantic connections (i.e. Causal links) are pre-computed in such a matrix to make Web service composition easier.

The causal link matrix controls all relevant Web services discovered by the Web services discover module. This control takes place in a semantic way according to the Matching functions (Table 2).

Example 7. (Causal link matrix illustration)

Consider the following matrix \(\mathcal{M}\) with coefficients in

\[
S_{W,s} = \{(S_a, S_b, S_c, S_d, S_e) \times [0, 1] \}
\]

\[
\mathcal{M} = \begin{pmatrix}
0 \{(S_a, 6)\} & 0 & 0 & 0 & 0 & \{(S_e, 1)\} \\
0 & 0 & 0 & 0 & 0 & \{(S, 1)\} \\
0 & 0 & 0 & 0 & 0 & \{(S_d, 6), (S, 1)\} \\
0 & 0 & 0 & 0 & 0 & \{(S_d, 6), (S_e, 1)\} \\
0 & 0 & 0 & 0 & 0 & \{(S_e, 6)\}
\end{pmatrix}
\]

The entry \(m_{5,3}\) (i.e. \(m_{\text{Warning Level}, \text{Organization}, \text{Table 3}}\)) is equal to \(\{(S_d, 6), (S_e, 1)\}\). Indeed a Web service \(S_3\) (Table 1) with one input parameter \(\text{Warning Level}\) and an output \(\text{Emergency Dpt}\) is semantically close to an abstract Web service \(S\) with an input parameter \(\text{Warning Level}\) and an output parameter \(\text{Organization}\) (Table 3). This semantic proximity is valued by the \(M_{\text{exact-aff}}\) match, i.e. \(\text{Sim}_T(S, S_3)\) (Table 2).

Moreover a Web service \(S_e\) with one input parameter \(\text{Warning Level}\) and an output \(\text{Organization}\) is semantically close to an abstract Web service \(S\) with an input parameter \(\text{Warning Level}\) and an output \(\text{Organization}\). This semantic proximity is valued by the \(M_{\text{in-out}}\) match, i.e. \(\text{Sim}_T(S, S_e)\) (Table 2).

Two Web services \(s_x, s_y \in S_{W,s}\) will be semantically composed as \(s_y \circ s_x\) in case a causal link matrix \(\mathcal{M}\) exists such that \(s_x \subseteq m_{i,j}\) with \(c_j \in \text{Input}(s_y)\).

This process follows an iterative backward chaining method from goals (as the output parameter of \(s_y\)) to an initial knowledge base, described as instantiated concepts.

5.3. Causal link matrix construction

A “set of similar Web services” of \(s\) is necessary to build the causal link matrix. Such a set, i.e. \(E_{ss}(s(r_i, c_j))\) refers to Web services semantically stored by the entry \(m_{i,j}\) (row \(r_i\) and column \(c_j\)) of a causal link matrix \(\mathcal{M}\). All causal links between each Web service of a set \(S_{W,s}\) are built according to the “set of similar Web services” \(E_{ss}(s)\). \(E_{ss}(s)\) will contain only Web services \(s_k\) with a strictly positive similarity with \(s(r_i, c_j)\) (Table 2).

Definition 4. (Set of similar Web services to \(s\): \(E_{ss}(s)\))

Let \(E_{ss}(s)\), the set of Web services similar to \(s\) defined by

\[
E_{ss}(s) = \{s_k \mid \text{Sim}_T(s(r_i, c_j), s_k) > 0\}
\]

with \(r_i \in \text{In}(s)\), \(c_j \in \text{Out}(s)\) and \(s \in S_{HWS}\) (5)

Example 8. According to the set \(\{S_a, S_b, S_c, S_d, S_e\}\), the \(E_{ss}(s)(S_b)\) set contains one Web services \(S_b'\) (In Organization, Out Physician, Pre\(S_b\), Post\(S_b\)). However a Web services \(S_{W'}'(\text{In Medical Dpt}, \text{Out Surgeon}, \text{Pre}_{S_{W'}}S_{W'}, \text{Post}_{S_{W'}}S_{W'})\) might be in the \(E_{ss}(s)(S_b)\) set in case Medical Dpt \(\subseteq\) Organization, Physician \(\subseteq\) Person and Surgeon \(\subseteq\) Person. The Web services \(S_{W'}\) and \(S_{W'}\) could be regarded as substitute Web services of \(S_b\).
The Algorithm 2 presents a primitive construction of the causal link matrix $M$. Such a matrix links each relevant Web service according to their input and output parameters. $M$ refers to a formal representation of all abstract possible pathways between Web service parameters (outputs, inputs). The Algorithm 2 complexity is related to two steps. Step 1 is function of $\bigcup_{i=1}^{n} Input(s_i)$ size whereas step 2 is function of $\bigcup_{i=1}^{n} Input(s_i) \cup \beta$.

**Algorithm 2: Causal link matrix construction.**

**Input:** $S_{W}$, $T$, $KB$, an empty Causal link matrix.

**Result:** Causal link matrix $M$.

```
begin
    foreach input $r_i \in \bigcup_{k=1}^{n} Input(s_k)$ do
        foreach input $c_j \in \bigcup_{k=1}^{n} Input(s_k) \cup \beta$ do
            if $c_j \in KB$ then $m_{r_i,c_j} = (KB, 1)$;
            else if $E_{ws}(s(KB, c_j)) \neq \emptyset$ then
                foreach $s^* \in E_{ws}(s(KB, c_j))$ do
                    if $r_i \in Input(s^*)$ then
                        $m_{r_i,c_j} = m_{r_i,c_j} \cup (s^*, Sim_T(s(KB, c_j), s^*))$;
            end
        end
    end
    return $M$;
end
```

The causal link matrices construction is function of the cardinality of output and input parameters of Web services in $S_{W}$. Suppose $\#(S_{W})$, $\#(Input(S_{W}))$ and $\#(Output(S_{W}))$ be respectively the number of Web services in $S_{W}$, the cardinality of input parameters of Web services in $S_{W}$ and the cardinality of output parameters of Web services in $S_{W}$. The algorithmic complexity for the causal link matrix construction is $\Theta(\#(Input(S_{W})), \#(Output(S_{W})))$ or $\Theta(\max(\#(Input(S_{W})), \#(Output(S_{W}))))^3$ so cubic in the worst case.

**Example 9.** The causal link matrix for $S_{W}$ with $KB \equiv \{Device.Address, BP\} = M_0$:

$$
\begin{pmatrix}
{(Add,1)} & {(BP,1),(S_{A},6)} & 0 & 0 & 0 & 0 \\
{(Add,1)} & {(BP,1)} & 0 & 0 & 0 & \{S_{A},1\} \\
{(Add,1)} & {(BP,1)} & 0 & 0 & \{S_{A},1\} & 0 \\
{(Add,1)} & {(BP,1),(S_{A},6)} & 0 & 0 & 0 & 0 \\
{(Add,1)} & {(BP,1)} & \{S_{A},6\} & \{S_{A},1\} & 0 & 0 \\
{(Add,1)} & {(BP,1)} & \{S_{A},6\} & \{S_{A},1\} & 0 & 0 \\
\end{pmatrix}
$$

In this section the causal link matrix has been introduced in order to prepare a Web service composition at functional level. The construction of such a matrix requires the knowledge of similarity functions $Sim_T$ (Table 2).

5.4. Causal link matrix issues

The key contribution of the Causal link matrix is a formal and semantic model to control a set of Web services which are relevant for a Web service composition. Web services of $S_{W}$ are supposed to be relevantly discovered in a discovery process [35, 7]. Thus the set of Web services $S_{W}$ is closed in order to limit the dimension of the Causal link matrix. This model allows performance analysis of proposed plans with a concrete view of the composition background: causal links and their semantic dependency. The Causal link matrix aims at pre-chaining Web services according to a semantic similarity based on their Output/Input specification. Thus the CLM describes all possible interactions as semantic connections between all the known Web services in $S_{W}$. The Causal link matrix is able to prepare a suitable context for an AI planning problem [25, 16, 28] with the purpose of obtaining complete, correct, consistent and optimal plan.

A set of ontologies $T$, a set of Web services $S_{W}$, a goal $\beta$, a knowledge base $KB$ and a semantic similarity function $Sim_T$ are required in order to satisfy such a challenging solution. With a terminology $T$, we deal with conceptual analysis (inference problems) and knowledge representation. A set of Web services refers to a set of actions for a planning problem. $\beta$ informs about plan directions (as searching concepts). A knowledge base $KB$ informs about initial conditions (instantiated concepts). Finally the similarity function $Sim_T$ semantically compares two parameters as concepts in $T$.

5.5. Web service composition through AI planning

The planning problem is formalized as a triple $\Pi = (S_{W}, KB, \beta)$. $S_{W}$ refers to a set of possible state transitions, $KB$ is an *Initial state* and $\beta \subseteq T$ is an explicit goal representation. The Web service composition method consists of finding a plan that produces the desired outputs $\beta$ according to a knowledge base $KB$. The causal link score allows the early detection of impossible, feasible and best links between Web services (Definitions 1 and 2). That is why our method is based on the causal link validity between Web service. The CLM of a specified domain allows to detect all Web service composition with semantic connections.

[22] propose a simpler form of AI planning in order to avoid problems [34] from planning-based Web services composition, e.g. non determinism and implicit goal. The set of Web services $S_{W}$ (i.e. Actions) is closed by assumption and the goal set $\beta$ refers to a set of concepts in a terminology $T$. Thus the proposed solution is well-defined: goals are explicitly given, initial state is well defined and Web services are strictly defined at functional level. So non determinism, implicit goal, fuzzy Web service description and behaviour are out of the question. Therefore it does seem possible to directly apply current AI planning methods to our specific problem.
The composition process consists of a recursive and regression-based approach. A Web service with a goal $\beta$ as output parameter has to be found in $S_{Ws}$. In case of success, the process is iterated with its input parameters as new goals. Alternatively, the process is stopped and the plan is reduced to the empty plan. All the process is recursive until all goals and new goals are concepts in $KB$ (stop condition). [22] presents the complete process of composition and returns a plan composed of valid causal links. Causal link matrices ease the regression-based search because all Web services are semantically well ordered in a robust and formal model. The solutions are plans wherein Web services are semantically chained by causal links. Instead a regression-based approach, other problem-solving techniques may be applied such as forward-chaining - called heuristic reasoning - may be applied [18].

Example 10. Suppose we are looking for a person responsible ($\beta \equiv \text{PersonID}$) for a specific WarningLevel ($KB \equiv \text{WarningLevel}$). The problem is formalized as $\Pi = \langle S_{Ws}, KB, \beta \rangle$ with $S_{Ws} = \{ S_a, S_b, S_c, S_d, S_e \}$. According to the causal link matrix of the domain $M_0$, a Web service composition contains two solutions according to the formal model, i.e. $S_b \circ S_d$ and $S_b \circ S_e$. Indeed the output parameter $\text{EmergDpt}$ of $S_d$ and an input parameter $\text{Organization}$ of $S_b$ match according to $M_0$. In the same way the output parameters $\text{Organization}$ of $S_c$ and the input parameter $\text{Organization}$ of $S_b$ match.

Even if we focus on a simple Web service composition for a better understanding, we may also consider more complex composition model including sequence, non determinism choice and parallel constructs.

6. Related work

Two different approaches [12, 15] propose matrices to represent the Web services domain. [12] solve an AI planning problem where actions are viewed as tasks. Actions are formally described with Preconditions and Effects. These tasks are executed by concrete Web services, according to a service/task (row/column) matrix. [15] propose a simple method to store Web service according to an input/output (row/column) matrix. The Matrix model used in [12, 15] do not propose reasoning about those matrices. In fact, such the matrices are simply considered as representation models. Moreover no semantic feature is introduced in their models.

Main functional level composition approaches compare Web services by explicitly matching their signatures descriptions: $\text{Sim}_{\text{W}}(s_y, s_x)$. This process is often called a Web service matchmaking problem. According to Web service capabilities, a service request matches a service advertisement if the request provides all the inputs (possibly more) needed by the advertisement while the advertisement generates all the outputs (possibly more) needed by the requester. A prominent representative model is the Semantic Matchmaker developed by [26]. Their algorithm operates on OWL-S descriptions and tests if the request can provide all required inputs and if the offer’s output satisfies the requester’s demands. The latter match degree may be weakened because an exact match is very strict and restrictive: thus an output (respectively input) $\text{out}_q$ of a request is also matching exactly to an offer’s output $\text{out}_o$ (respectively input) if $\text{out}_q$ is a direct subclass (respectively direct super-class) of $\text{out}_o$. Moreover exact match, plug-in and subsumes match [26] inspired by Software Engineering [36], are introduced to overcome in some way the limitations of a matching approach based on exact match. Nevertheless subsumption and satisfiability as standard inference are not sufficient for solving inference problems [21]. In the case of a non-standard match, an intersection [24], a partial (Contraction) or potential (Abduction) [13] ranking can be used to match a service request and offer. In [32], a similar approach is pursued. The issue of approximate matches, to be somehow ranked and proposed in the absence of exact matches, is discussed in [13, 21].

Authors of [37] propose a forward chaining approach to solve a planning problem. Their composition process terminates when a set of Web services that matches all expected output parameters given the inputs provided by a user is found. [25] describes the semantics of processes and their inputs, outputs, preconditions and effects as axioms in situation calculus. These axioms are mapped onto Petri net representations, which then describe the execution of Web service control constructs.

7. Conclusion and future work

In this paper we outlined the main challenges faced by semantic Web Services. Furthermore, we showed how the causal link matrix tackles this challenge by providing a formal model that allows automated Web service composition. Finally a functional level based Web Service architecture takes advantage of OWL-S or WSMO specification to support automatic composition between services. Despite the fact that Web service composition is in its infancy, some proposals are studied to solve such a problem at functional level. Nevertheless no formal model is proposed to help the automation of composition at the best stage of our knowledge. A causal link matrix as model for functional level composition is proposed in order to obtain a robust, secure, and verifiable model [34]. Contrary to [12, 15], our matrix model pre-computes the semantic similarities between Web services (individual inputs and outputs) in order to make Web service composition easier in a close set of Web service. CLMs not only allow to bound the Web service con-
text but also provide a semantic context for the Web service composition. Thus Web service composition is viewed as causal link composition wherein the composition plan is built from a simple causal link matrix analysis [23].

For further studies we plan to improve and extend the semantic Web service matching functions set for optimization reasons. We are keeping in mind that process level composition as well as choreography and orchestration cycles are main issues in order to obtain a correct composition model.

References