A High-Level Functional Matching for Semantic Web Services

Islam Elgedawy, Zahir Tari, and James A. Thom

School of Computer Science and Information Technology,
RMIT University, Melbourne, Australia
{elgedawy, zahirt, jat}@cs.rmit.edu.au

Abstract. Existing service matching techniques such as keyword-based and ontology-based, do not guarantee the correctness of the matching results (i.e. do not guarantee fulfilling user goals). This paper deals with this problem by capturing the high-level functional aspects (namely goals, contexts, and expected external behaviors) for both web services and users in a machine-processable format, then matching these aspects using the proposed functional substitutability matching scheme (FSMS). Based on FSMS, this paper describes a direct matching technique in which a user request is examined against one service description at a time, such that web services match users requests when they have substitutable goals, contexts and expected external behaviors. The substitutability semantics between the elements of application domains are captured via the proposed substitutability graphs, which are used during the matching process to mediate between users requests and web services descriptions. Simulation results show that the proposed matching approach succeeds in retrieving only the correct answers, while keyword-based and ontology-based retrieval techniques could not eliminate the appearance of false negatives and false positives.

1 Introduction

Existing matching techniques used in web service discovery, such as keyword-based techniques and ontology-based techniques, fail to provide high matching precision [4]. These techniques can be classified as "generic" as they are supposed to work in all contexts and for all application domains. In a nutshell, generic matching techniques examine the descriptions of web services by eliciting the various concepts, including inputs, outputs and entities, from the descriptions. Later, such concepts are matched using a keyword-based approach ([3]), a more precise approach (which uses generic subsumption rules given via domain taxonomies [8]), a form of Logic ([1, 9]), or a combination of approaches. Generic matching techniques are not suitable for web services as they do not take into consideration additional semantics related to web services and users[1], such as goals, contexts and expected external behaviors, that are needed to obtain correct results [4, 6, 7], as they provide information about (what a service

[1] The term "user" is used to refer to humans and machines.

B. Benatallah, F. Casati, and P. Traverso (Eds.): ICSOC 2005, LNCS 3826, pp. 115–129, 2005.

does/what a user wants) [4], the adopted constraints [6] and how the required goal is going to be achieved[7]. To guarantee the correctness of the matching results, we have identified the following requirements:

- User goals, contexts and expected external behavior should be semantically captured in a machine understandable format so that the matchmaker can understand them and later use them to find the correct answers.
- The high level functional aspects of web services, such as goals, contexts and expected external behavior should be explicitly captured in a machine understandable format so that the matchmaker can understand them and use them to find the correct answers.
- The functional semantics of the application domain should be explicitly captured in a machine understandable format such that the matchmaker can use them to mediate between the user request and web services descriptions.
- A formal matching scheme should use all these captured semantics (user semantics, web service semantics, and application domain semantics) and should guarantee the correctness of the matching results.

The G^+ model we previously proposed in [4, 6] captures high-level functional aspects (such as goals, functional contexts[2], and expected external behaviors) for both web services and users. Existing solutions for describing web services (such as OWL-S[2]) do not have explicit representations for these high-level functional aspects as they are described via text descriptions in the service profile. Recently, WSMO [9] followed a similar approach to ours by providing an explicit semantic representation for the high-level functional aspects such as goals, however it lacks explicit representation for web services behaviors (both internal and external).

The matching scheme indicates what comparison aspect between the involved elements is used, what matching rule is used, and how to judge the correctness of the matching results. Therefore, this paper introduces the functional substitutability matching scheme (FSMS) that uses high-level functionality as the comparison aspect, substitutability as the matching rule, and goal achievement as the correctness criterion.

The proposed direct matching approach starts by filtering services that correspond to different application domains and supports different domain roles. Then checks the substitutability of the G^+ models against user G^+ model to find the suitable set of services that fulfill user request. Substitutability between two G^+ models is determined according to the substitutability status between the functional contexts and the operation sequences of the corresponding scenarios. Functional context substitutability is determined according to the substitutability of their pre, post and describing sets of constraints. Substitutability status between two sets of constraints is determined by finding a sequence of consistent transformations that transform the elements of the source set into the elements of the target set using the proposed substitutability graphs. Substitutability of

[2] A functional context describes the requirements for correct goal achievement, as indicated in later sections.

operation sequences is determined according to the substitutability of their corresponding behavior models. A behavior model is a sequence of states elicited by tracing the transition points between the operations in the corresponding operation sequence, such that a state is represented by the active constraints at the corresponding transition point. Two states are matched according to the substitutability status between their constraints.

This paper is organized as follows. Section 2 provides an overview of the G^+ model used to capture the high-level functional aspects. An overview of the meta-ontology used is also provided. Section 3 introduces the functional substitutability matching scheme. Section 4 provides details related to the proposed direct matching approach between two given G^+ models. Experiment results are described in Section 5, and Section 6 concludes the paper.

2 The G^+ Model

The G^+ model [4, 6] is an extended goal model that provides an integration of various concepts, including operational goals[3], the corresponding functional contexts[4], and the corresponding realization scenarios[5]. A functional context is represented by three sets of constraints[6]: pre-constraints (goal pre-conditions), post-constraints (goal post-conditions), and describing-constraints. Table 1 provides an example of a G^+ instance, describing HotelReservationService, where Hotel-Reservation, Submit-Room-Details, Submit-Payment-Details, Send-Confirmation are tourism domain operations defined in the used domain ontology.

Table 1. A G^+ Example

Goal=	"Hotel-Reservation"
FunctionalContext =	⟨ {CreditCard.Type = VISA, Hotel.Country = Australia},
	{Payment.Status = OK, Confirmation.Status = Sent},
	{Hotel.Function ∈ (Casino, Bank, Baby Setting, SPA, GYM)}⟩
RealizationScenario=	SubmitRoomDetails: SubmitPaymentDetails: SendConfirmation

As an application domain can be described by multiple ontologies, our approach uses a meta-ontology that acts as a schema for domain ontologies that indicate what should be captured in an ontology and how. Hence during the matching process, specific types of application domains' elements and their semantics will be used. The proposed meta-ontology consists of two layers: schematic layer and a semantic layer. At the schematic layer, the types of the domain elements are defined. At the semantic layer, the relations between the domain elements are

[3] A goal that is represented by an application domain operation.
[4] The requirements for correct goal achievements.
[5] A scenario represents a sequence of application domain operations that tells one story about how to achieve the goal.
[6] Any constraint is formulated as Entity.Attribute Operator Value.

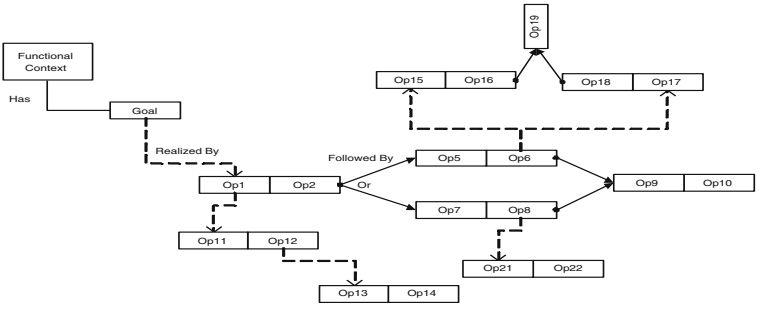

Fig. 1. An Example of a Scenario Network

captured. The proposed approach, however, restricts these relations to only one type, namely the functional substitution relation, as FSMS uses substitutability as the matching rule. The components of the schematic layer are concepts, operations and roles. Concepts represent the domain entities, which are described by a set of features. A feature is an attribute with its corresponding value. Operations represent the domain legitimate transaction types. Every operation is described by a set of features. Every operation has a set of input concepts and a set of conditions over these concepts. Every operation has a set of output concepts and a set of conditions over these concepts. Roles represent domain legitimate actors. Every role is described by a set of features.

As many different scenarios can describe the achievement of a given goal, a more complex G^+ model can be defined by a scenario network allowing multiple abstraction levels, showing how such a goal could be achieved. An example is given in Figure 1. Tracing a goal scenario network from the goal node until a leaf operation node represents one of the expected paths for achieving the goal. Such a path is called *Goal Achievement Pattern* (GAP). A GAP describes a global (end-to-end) snapshot of how a service goal is expected to be accomplished. This snapshot provides information that helps the matchmaker to anticipate the external behavior of a service in a given context. In a nutshell, a GAP consists of the following information: a functional context, a goal, and an operation sequence. A functional context represents the context of the corresponding service defined in conjunction with any existing sub-contexts (that is the value cases of the branching conditions along the path). Sub-contexts are added to the set of pre-constraints that belongs to the functional context. An operation sequence is a result of the tracing of a scenario network from the goal node till a leaf operation node. Formally, $\mathsf{GAP} = \langle \mathsf{Cntxt}, \mathsf{G}, \mathsf{OpSeq} \rangle$, where Cntxt is the GAP functional context, G represents the GAP goal, and OpSeq is the GAP operation sequence.

A scenario network provides multiple abstraction levels for describing the achievement of a given service goal (that is, an operation in a given scenario could be described by another scenario network). So, we can extract multiple abstraction level GAPs such that a group of GAPs will describe the same story but at different levels of detail. This provides great flexibility to the matchmaker to choose a suitable abstraction level to work on.

3 The FSMS Matching Scheme

The following factors are used to judge the correctness of the results produced by a the proposed matching approach. (i) A comparison aspect that indicates which facts (about the involved entities) will be considered. (ii) A matching rule that indicates how the comparison aspect will be examined. Finally, (iii) a correctness criterion that validates the obtained results after applying the matching rule. Any matching scheme should define its own comparison aspect, matching rule and correctness criterion. This section describes a new matching scheme, called *Functional Substitutability Matching Scheme* (FSMS), which uses *functionality* as the comparison aspect, *substitutability* as the matching rule, and *goal achievement* as the matching correctness criterion. A service's functionality is the service's capability for achieving a given goal. This is captured using the G^+ model. In general, a goal G is considered achieved when transforming a set of constraints W_i into another set of constraints W_j. This is denoted as $(W_i \longmapsto_G W_j)$, and its reads as "the goal G is achieved when transforming W_1 into W_2 using S", where $\longmapsto$ is the achievement operator. Adopting the G^+ model, a goal is considered achieved when transforming the pre-constraint set into the post-constraint set of the functional context using the defined GAP (that is invoking the corresponding sequence of operations).

A user specifies both the request (in the G^+ format) and the G^+ models to be published. The matchmaker succeeds in the matching of a service with the user's request when the service can achieve the request goal. For example, if the request's goal G is achieved when $(W_1 \longmapsto_G W_2)$, and a service S achieves G by transforming W_3 into W_4 $(W_3 \longmapsto_G W_4)$, then S is considered a match for the request when $W_1 \Rightarrow W_3$ (W_1 can substitute W_3) and $W_4 \Rightarrow W_2$ (W_4 can substitute W_2). This enables the transformation of W_1 into W_2 by invoking S, which means G is achieved. Constraint substitution via implication is restrictive as it leads to the appearance of false negatives. For example, the constraint (City.Name=Cairo) implies the constraint (Country.Name= Egypt). However this cannot be derived using implication as the two constraints have two different scopes. To overcome this problem, we propose the concept of "constraints semantics subsumption" that adopts the substitution semantics of application domains; to find a transformation that transforms the source constraints into the target constraints to determine the substitutability status between these constraints. We propose to extend the meta-ontology to support use of substitutability graphs.

Definition 1. *(Substitutability Graph) A substitutability graph is a directed graph, where a node is in the form of entity.attribute and an edge indicates the substitutability direction. Every edge has the corresponding substitutability conditions that must be satisfied in order to substitute a given entity's attribute with another entity's attribute* ***with respect to*** *a given domain operation. Also every edge has the corresponding conversion function between the attributes' values.*

As an entity could be a concept or a role, this implies for every domain two substitutability graphs will be defined: a concept substitutability graph and a

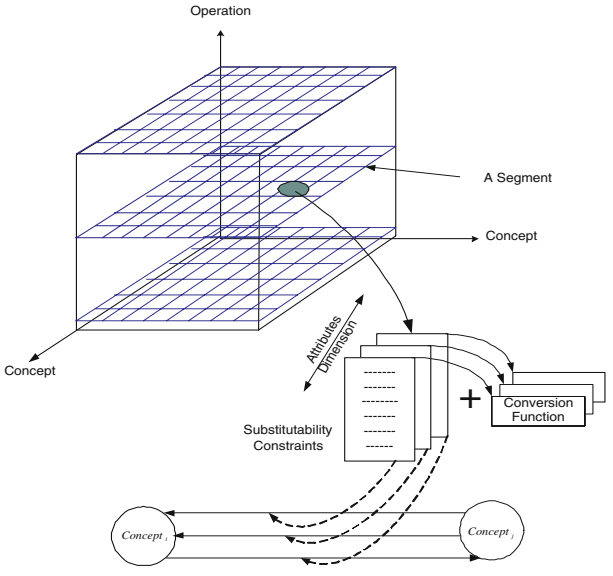

Fig. 2. Concepts Substitutability Graph

role substitutability graph. Figure 2 shows how a concept substitutability graph has been introduced: for every domain operation, and for every pair of concepts, substitutability conditions are defined.

If no attributes are explicitly defined; this means all the concepts' attributes with the same name are substitutable with respect to such domain operation. Their corresponding conversion functions will be a simple equality between the attributes values. However if only specific attributes are substitutable, for every pair of substitutable attributes an edge should be created such that the corresponding substitutability constraints and conversion functions are defined. For example, the attribute Rank of the concept Hotel can have the values in the set {5,4,3,2,1}, which corresponds to the number of stars of the hotel, while the attribute Class of the concept Accommodation can have values in {Business,Economy}. For a hotel-reservation operation in the tourism domain, the concept Accommodation could be substituted with the concept Hotel if the accommodation type is temporary. Hence, the attribute Class could be substituted by the attribute Rank using a specific conversion function, as indicated in Table 2.

Table 2. An Example of a Substitutability Graph Entry

Source	Target	Conversion Function	Substitutability Conditions

Hotel.Rank	Accommodation.Class	If Hotel.Rank $\geq$ 3
		Then Accommodation.Class = Business
		Else
		Accommodation.Class = Economy
		End IF
			Accommodation.Type=
			Temporary

For instance (Hotel.Rank=4) and a service description (that indicates that the service can book temporary business class accommodations), the matchmaker can return the service as a correct one if the other constraints in the request are fulfilled. Formally, we introduce **concepts substitutability graph** SG_c as a directed graph such that $SG_c = \bigcup_{i=1}^{n}\{\langle Op_i, V_c, E_c\rangle\}$, where Op_i is a domain operation, n is the number of operations in the domain, V_c is a set of graph nodes such that $V_c \subseteq \Delta_c \bigotimes \Delta_A$ that Δ_c is the set of all concepts Δ_A is the set of all attributes. E_c represents a set of graph edges such that $\forall E_i \in E_c, E_i = \langle V_a, V_b, \Pi_{ab}, \Psi_{ab}\rangle$. Π_{ab} is the set of substitution conditions that must be satisfied in order for the concept.attribute represented by the node V_a (source node) to be substituted with the concept.attribute represented by the node V_b (target node)[7]. Ψ_{ab} is the conversion function that maps the value of the source node into a value for the attribute in the target node. In a similar way, we define **role substitutability graph** SG_r. A substitutability graph is used to determine if there exists a direct substitution (DS) between constraints' scopes.

Definition 2. *(Direct Substitution) Given two constraints Cn_i and Cn_j and two scopes A and B, such that A is the scope of Cn_i and B is the scope of Cn_j, a goal G, and a set of constraints W such that $Cn_j \in W$, there exists a direct substitution (DS) between A and B with respect to G and W iff there exists a valid path P=$L_1, L_2, ..., L_n$ between A and B in the corresponding substitutability graph such that $W \Rightarrow \wedge_{i=1}^{i=n}\Pi_{L_i}$ where L_i is an edge between two scopes, and Π_{L_i} is the substitution constraints of edge L_i*[8].

When a matchmaker attempts to transform a constraint Cn_i (the source constraint) into another constraint Cn_j (the target constraint) with respect to a given goal, first it checks if there exists a DS between their scopes. If so, it uses the conversion function(s) to transform the source constraint Cn_i into another constraint Cn_k such that $Cn_k \Rightarrow Cn_j$[9]. Otherwise the transformation is considered not valid and the constraints cannot be substituted.

Two scopes are considered *reachable* if there exists a DS, or a sequence of DSs between them. This sequence of DSs is the **required transformation** in order to substitute the target constraint with the source one. When such a transformation exists, the source constraint is considered to semantically subsume the target constraint with respect to the involved goal.

Definition 3. *(Constraints Semantic Subsumption) Given two constraints Cn_i, Cn_j and a goal G, Cn_i semantically subsumes Cn_j with respect to G (denoted as $Cn_i \mapsto_G Cn_j$) iff Cn_i and Cn_j scopes are reachable with respect to G using a transformation β, and Cn_i is transformed to Cn_q using β such that $Cn_q \Rightarrow Cn_j$.*

[7] The correctness of the substitution conditions is the responsibility of the ontology engineer.

[8] A set of constraints is treated in implication as one constraint that is a conjunction of the set elements.

[9] It is important to note that Cn_k and Cn_j have the same scope.

Adopting Definition 3, Cn_i matches Cn_j when the involved transformation does not violate the set of active constraints existed at the substitution time.

4 The Direct Matching Approach

Matching two web services based on their high-level functional aspects using FSMS is determined according to the substitutability of their corresponding G^+ models. We will first show how simple G^+ models (represented by only one GAP) are matched. Later we will show the same process on complex G^+ models (represented by a GAP forest).

Definition 4. *(Direct G^+ Matching) Given two simple G^+ models G_i^+ and G_j^+, G_j^+ can be substituted by G_i^+, denoted as ($G_i^+ \trianglerighteq G_j^+$), iff ($Op_i = Op_j$) $\wedge$ ($Ctxt_i \trianglerighteq_{Op_j} Ctxt_j$) $\wedge$ ($GAP_i \trianglerighteq_{Op_j} GAP_j$), where Op_i and Op_j are the operations representing the goals of G_i^+ and G_j^+ respectively. $Ctxt_i$ and $Ctxt_j$ are the functional contexts of G_i^+ and G_j^+ respectively. GAP_i and GAP_j are the goal achievement patterns of G_i^+ and G_j^+ respectively.*

Definition 4 indicates that G_j^+ matches G_i^+ when they are represented by the same domain operation and the achievement requirements of G_j^+ (captured by its functional context) are substitutable by the achievement requirements of G_i^+. So, the goal achievement pattern of G_j^+ is substitutable by the goal achievement pattern of G_i^+. The first step to realize Definition 4 is to illustrate how two functional contexts are going to be matched in FSMS; later we will show how two GAPs will be matched in FSMS.

Definition 5. *(Functional Context Matching) Given two contexts $Ctxt_i$, $Ctxt_j$ and a given goal G, $Ctxt_j$ can be substituted by $Ctxt_i$ with respect to G (denoted as $Ctxt_i \trianglerighteq_G Ctxt_j$) iff (($Ctxt_i^{Pre} \trianglerighteq_G Ctxt_j^{Pre}$) $\wedge$ ($Ctxt_i^{Post} \trianglerighteq_G Ctxt_j^{Post}$)$\wedge$ ($Ctxt_i^{Desc} \trianglerighteq_G Ctxt_j^{Desc}$)).*

Definition 5 indicates that a functional context $Ctxt_j$ matches $Ctxt_i$ when its constraints sets (pre, post and describing) are substitutable by the corresponding constraints sets of $Ctxt_i$. More details about context matching can be found in [6]. So, two GAPs are matched in FSMS as indicated in Definition 6.

Definition 6. *(GAP Matching) Given two goal achievement patterns GAP_i and GAP_j such that $GAP_i = \langle\ Cntxt_i,\ G_i,\ OpSeq_i\ \rangle$ and $GAP_j = \langle\ Cntxt_j,\ G_j,\ OpSeq_j\ \rangle$. GAP_j can be substituted by (matches) GAP_i denoted as ($GAP_i \trianglerighteq GAP_j$) iff ($G_i = G_j$) $\wedge$ ($Cntxt_i \trianglerighteq_{G_j} Cntxt_j$) $\wedge$ ($OpSeq_i \trianglerighteq_{G_j} OpSeq_j$).*

Definition 6 indicates that matched GAPs will be realizing the same goal and will be having substitutable contexts and substitutable operation sequences. Operation sequences could be matched syntactically, semi-semantically (adopts 1-to-1 state matching) or semantically (adopts many-to-many state matching).

The semantic approach takes into consideration the effect of a group of operations on the external behavior of the service. The effect of invoking a sequence of operations on (the external behavior of) a service resembles a sequence of state transitions, where a state represents the (active) constraints at the corresponding transition point. Every state corresponds to a specific set of user interactions with the service (the external behavior). These user interactions are reflected by the active constraints captured in the state. The state model corresponds to the transition point between two operations (Op_x and Op_{x+1}) and differentiates between the active and idle constraints, as the idle constraints do not have any effects over the successor operation so they will be discarded during state matching.

Definition 7. *(State Definition) Given an operation sequence* $OpSeq = Op_0, Op_1, \ldots, Op_n$, *a state* S_x *between the operations* Op_{x-1} *and* Op_x *is defined as* $\langle f_x^e, f_x^i \rangle$ *such that* $f_x = f_x^e + f_x^i$, *where* f_x *is the set of active constraints at transition point* x, f_x^e *is the set of effective constraints,* f_x^i *is the set of idle constraints and* $+$ *represents the union operator between two sets.* f_x, f_x^e *,and* f_x^i *are computed as follows:(1)* $f_0 = Ctxt^{Pre}$. *(2)*$f_x = f_{x-1}^i + Op_{x-1}^{Post}$, *where* $1 \leq x \leq n+1$. *(3)* $f_x^i = f_x \diamond_G Op_x^{Pre}$, *where* $\diamond_G$ *is the semantic difference between two sets of constraints with respect to G.*[10] *(4)*$f_{n+1}^e = f_{n+1}$. *(5)* $f_{n+1}^i = \{\}$.

Every state has a corresponding scope that is defined as the set of element.attribute appearing as scopes in f_x^e of the state. The following example indicates how a state is automatically created.

Example 1. *(State Creation) Let us consider two consecutive operations Submit-Payment(Payment):Payment and Confirm-Order (Order, Payment): Order in a given purchase online transaction. The pre-conditions of "Submit-Payment" operations are* $\{Payment.method = Null\ ,\ Payment.details = Null\ \}$, *while its post-conditions are* $\{\ Payment.method \neq Null, Payment.details \neq Null, Payment.status = valid\ \}$. *The pre-conditions of "Confirm-Order" are* $\{Payment.status = valid, Order.status = created\ \}$, *while its post-conditions are* $\{\ Order.status = confirmed\ \}$. *According to Definition 7, the state that represents the transition point between "Submit-Payment" and "Confirm-Order" will be as follows: Assuming the independent set* $f_{x-1}^i = \{Order.status = created\}$, *hence* f_x *is equal to* $\{Order.status = created\ ,\ Payment.method \neq Null, Payment.details \neq Null, Payment.status = valid\ \}$, *the effective constraints are the ones that imply the pre-conditions of "Confirm-Order" operation and the rest of the constraints will be the independent idle constraints as follows:*

f_x^e	f_x^i
$\{Order.status = created,$ $Payment.status = valid\}$	$\{Payment.method \neq Null,$ $Payment.details \neq Null\}$

[10] A semantic difference between f_x and Op_x^{Pre} is a subset of f_x that has no reachable scopes to elements of Op_x^{Pre}.

The corresponding state sequence is created by applying Definition 7 at every transition point, by tracing all the transition points in a given operation sequence. After automatically constructing the state sequences from the involved operation sequences, they are going to be matched using FSMS such that when the two state sequences are matched, the corresponding operation sequences will be considered matched. State sequences will be matched adopting many-to-many manner, in which a group of states will be examined against another group of states.

Definition 8. *(State Matching) Given two states* $S_x = \langle S_x^e, S_x^i \rangle$, $S_y = \langle S_y^e, S_y^i \rangle$ *and a goal G,* S_y *can be substituted by* S_x *with respect to G (denoted as* $S_x \trianglerighteq_G S_y$*) iff* ($S_x^e \trianglerighteq_G S_y^e$).

As the many-to-many approach is adopted, state merging is required to realize such an approach. A state is represented by a set of constraints at a given transition point. Merging two states means generating a new state that represents the set of constraints resulted after forming a virtual composite operation, that is resulted from merging the operations following the transition points of the merged states, as indicated in Figure 3.

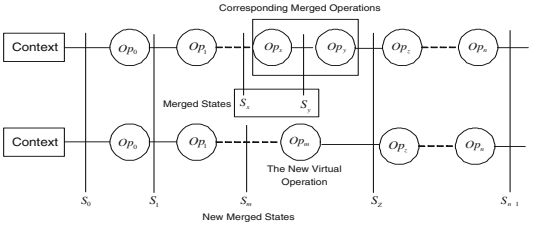

Fig. 3. Merging States

Op_m should not affect any other state in the corresponding sequence. To maintain this principle, both the pre/post conditions of Op_m are defined as follows: $Op_m^{Pre} = Op_x^{Pre} + (Op_{x+1}^{Pre} \diamond_G Op_x^{Post})$, and $Op_m^{Post} = Op_{x+1}^{Post} + (Op_x^{Post} \diamond_G Op_{x+1}^{Pre})$ [5]. The new state resulting from the merge S_m will be created according to the operator defined in Definition 7, adopting the values of Op_m's pre/post conditions (see Definition 9).

Definition 9. *(State Merge) Given two consecutive states* $S_x = \langle f_x^e, f_x^i \rangle$, $S_{x+1} = \langle f_{x+1}^e, f_{x+1}^i \rangle$ *and a goal G, the state* $S_m = \langle f_m^e, f_m^i \rangle$ *resulting from merging* S_x *and* S_{x+1} *with respect to G (denoted as* $S_m = S_x \oplus_G S_{x+1}$*) is defined as follows:* $f_m^i = (f_x^i \diamond_G f_{x+1}^e)$. $f_m^e = (f_x^i + f_x^e) \diamond_G (f_x^i \diamond_G f_{x+1}^e)$.

Definition 10. *(Expandable State) Given states* S_x *and* S_y *belonging to* $StatSeq_i$ *and* $StatSeq_j$ *respectively and a goal G,* S_x *is expandable with respect to* S_y *and G iff there exists a state* S_q *belonging to* $StatSeq_i$, $x \leq q$, *such that* ($S_w \trianglerighteq_G S_y$), *where* S_w *is a new state resulting from merging the states from* S_x *to* S_q.

Definition 10 implies the expansion direction is "down", however a state could be expanded in "up" direction, meaning that this will be merged with its predecessors. In order to realize the many-to-many matching approach, we need to determine both the states in a given sequence that should be expanded as well as the direction for the matching. We propose a transformation procedure, called

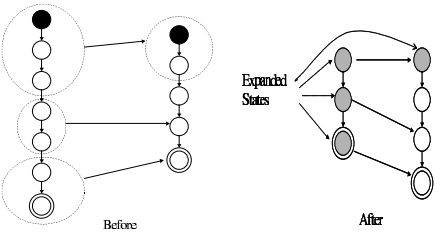

Fig. 4. Before and After Invoking SEQA

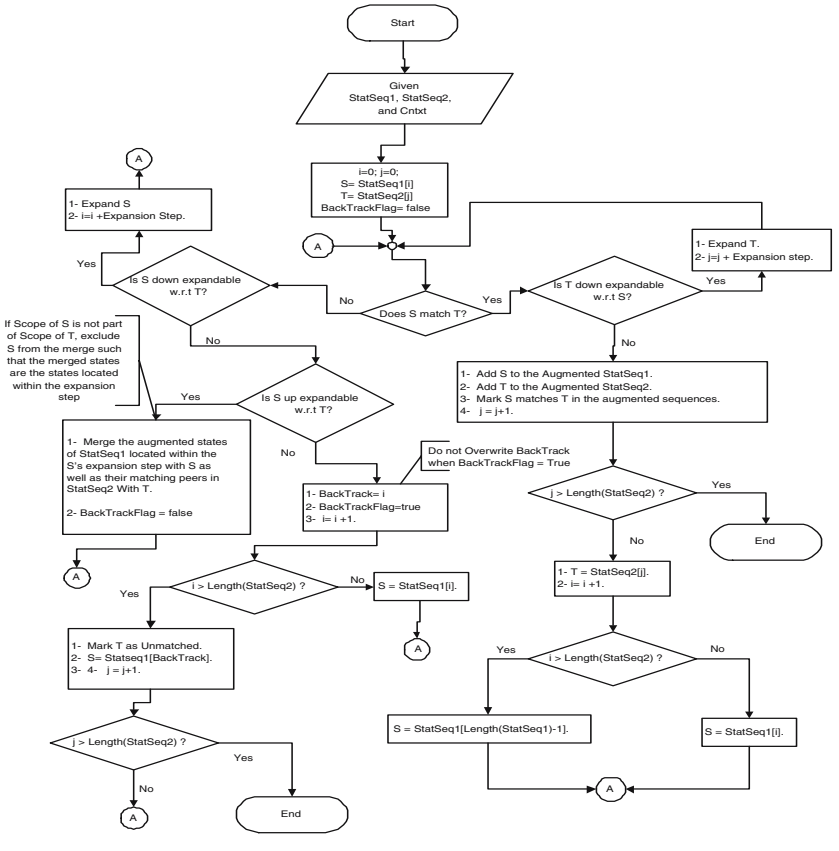

Fig. 5. SEQA Flow Chart

sequence augmenter (SEQA), to decide which and when states will be merged and the direction of merging. SEQA accepts two state sequences, a source sequence and a target sequence and the involved goal as its input, then returns two augmented state sequences and their corresponding matching peers. Figure 4 shows how the sequence augmenter works.

Figure 5 depicts the flow chart of SEQA. $StatSeq1$ is the source sequence and $StatSeq2$ is the target sequence. Let T represent the current target state to be matched in $StatSeq2$ and S represent the current proposed matching state in $StatSeq1$ that will be examined against T. Using Definition 8, SEQA checks whether or not S matches T. If this holds, this will mean that $Scope(S) \supseteq Scope(T)$, which gives an opportunity to check whether S also matches T's successor when merged with T. If this is true, T will be down expanded, and this process will be repeated until T is not down expandable. The current S is added to the augmented source sequence and the current T (the expanded T if so) is added to the augmented target sequence. Also, S and T are marked as a matching peer. S may not match T, meaning either they have different scopes or they have the same scope but the states' conditions contradict, meaning they will never match. However, having different scopes implies either (Scope(S) $\subset$ Scope(T)) or (Scope(S) $\cap$ Scope(T) = $\emptyset$).

When Scope(S) $\subset$ Scope(T), there is a chance for S to match T by down expanding S. Even for the case of (Scope(S) $\cap$ Scope(T)= $\emptyset$) by down expanding S, there may be a chance for the successors of S to match T. Hence, SEQA will try to down expand S, marked as backtracking point. If S down expansion fails, there is an opportunity for S to be up expandable with respect to T. If up expansion succeeds, then the augmented sequences (source and target) should be restructured. This happens by both merging all the states that lie within the expansion step and merging their matching peers in the other augmented sequence. If the up expansion procedure fails, this means the current S cannot match the current state T. Therefore SEQA will try the S's successor to match it with T and also will check all the previous scenarios. If one of the previous scenarios works and S matches T, SEQA prepares the next state in the target sequence to be matched. But if all the previous scenarios did not work for all the successors of S, this implies that T cannot be matched with the given source state sequence. Therefore T will be marked as unmatched, and SEQA prepares the successor of T to be matched. It backtracks to the S's state that first tested with the unmatched T so as to give a chance for the successor of T to be checked against this backtracked S. SEQA considers a source state sequence is a match for a target state sequence, when every state in the target augmented sequence must be substituted by an augmented state in the source in an order-preserving manner. Complexity of SEQA is $O(n^3)$. The complexity of SEQA is high as the worst case is every state in the target sequence must be examined against the states of the source sequence for up and down state expansion, which costs $O(n^2)$.

5 Validation

The correctness of matching results can be judged through the F-measure metric [11], as when having its value equal to one implies the matching results are totally correct. Unfortunately, there are neither benchmarks nor standard data sets for matching semantic web services. Hence we opted to use a random approach, where a random data set and queries are generated so as to validate the devised matching techniques, following the same simulation experiments indicated in [6] used for testing context matching. This section will therefore focus on the evaluation of devised GAP matching techniques and compare them to the semantic, semi-semantic and syntactic approaches.

Work-Load Generation. The proposed technique requires the existence of a domain ontology that adopts the meta-ontology structure. The elements of the conducted experiments are: a domain ontology (that includes concepts, operations, roles and the substitutability graphs); a data set of generic GAPs; and a query set and its correct answers. We have generated a random number of concepts, a random number of operations and a random number of roles to represent the domain elements. To make sure there are no contradicting conditions when a new operation is generated, the following restrictions are followed when generating the domain operations: (i) every operation has one distinct concept as input parameter and it will be the operation output concept as well. (2) Every operation has one pre condition over an attribute of the input concept, such that the value of this attribute equals to its lower limit. (3) Every operation has one post condition over the same attribute used in the pre conditions such that the value of this attribute is less than its upper limit. Operation sequences are generated by selecting a random number of operations from the generated operations; in order to form our data set of generic GAPs.

Experiment Logic. We have generated a random data set of generic GAPs. Then a query set is constructed by randomly selecting 10% of the data set. Experiments are performed as follows. First, data and query sets are generated. Second, ten mutated query sets are produced such that the first mutated query set has the first 10% of the query set being mutated, the second mutated query set has the first 20% of the query set being mutated, and so on until the tenth mutated query set has 100% mutated queries. Third, all query sets are applied to the semantic matching approach, the semi-semantic matching approach and the syntactic approach. Fourth, the retrieval precision is calculated as indicated before for all approaches. Finally, the above procedure is repeated 1000 times and the final average result is computed.

Without a loss of generality, the mutation process is performed by merging all the operations of a given GAP into one operation such that the new mutated GAP will have only one operation. This operation is constructed as follows: (1) Its input is a collection of the GAP operations' inputs. (2) Its output is a collection of the GAP operations' outputs. (3) Its pre condition is a conjunction of

[11] F-measure $= \frac{2 \times Precision \times Recall}{Precision + Recall}$.

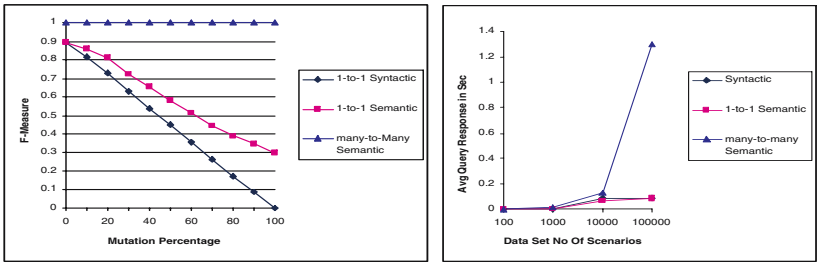

Fig. 6. A Comparison of GAP Matching Techniques

the GAP operations' pre conditions. (4) Its post condition is a conjunction of the GAP operations' post conditions. The pre/post conditions of the new constructed operation are deliberately mutated to generate new sets of conditions, so that the scopes of the original conditions are reachable from the scopes of the mutated conditions using a randomly generated substitutability graphs. For example, constraint (x>12) will be mutated into constraint (y>12) such that x is reachable from y. Experiment results are shown in Figure 6.

As the original query set and the mutated query sets have the same answers, the syntactic approach fails to answer the mutated queries. This is reflected by the decrease of retrieval precision, as the percentage of mutated queries increases. Also the semi-semantic approach fails to answer the mutated queries, except for the cases that its GAP has only one operation. Therefore, the semi-semantic approach could have the same behavior of the syntactic approach against the mutated queries. This indicates that many-to-many matching approach should be used instead of one-to-one approach when semantic matching is adopted. However, precision is expensive as indicated in the figure but we believe there is a good potential for performance enhancement as basic retrieval techniques are used in these experiments.

6 Conclusion

This paper demonstrates that capturing the semantics of web services, users and application domain in a machine-processable format is crucial for obtaining correct matching results. This paper proposed an advanced matching scheme for semantic web services, called *Functional Substitutability Matching Scheme* (FSMS), which uses high-level functionality (as a comparison aspect), substitutability (as a matching rule), and goal achievement (as a correctness criterion). The application domain functional substitutability semantics are captured via concept and role substitutability graphs. Adopting FSMS, we devised a direct matching technique for semantic web services that is shown to provide correct matching results (more details about other approaches are in [10]). Aggregate service matching adopting FSMS is the future extension of this work (more details in [5]).

Acknowledgment

This project is proudly supported by the ARC (Australian Research Council), under the ARC Linkage project no. LP0347217.

References

1. J. Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In *Proceedings of Workshop on Application of Description Logics*, Austria, September 2001.
2. OWL Services Coalition. Owl_S : Semantic markup for web services. http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2003.
3. X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 132–143. Morgan Kaufmann, 2004.
4. I. Elgedawy. A conceptual framework for web services semantic discovery. In *Proceedings of On The Move (OTM) to meaningful internet systems*, pages 1004–1016, Italy, 2003. Springer Verlag.
5. I. Elgedawy and Z. Tari. Aggregate high-level functional matching for semantic web services. Technical Report TR-05-3, RMIT University, Australia, 2005.
6. I. Elgedawy, Z. Tari, and M. Winikoff. Exact functional context matching for web services,. In *International Conference on Service Oriented Computing (ICSOC)*, November 2004.
7. I. Elgedawy, Z. Tari, and M. Winikoff. Scenario matching using functional substitutability in web services. In *Proceedings of the International Conference on Web Information Systems Engineering (WISE)*, 2004.
8. P. Ganesan, H. Garcia Molina, and J. Widom. Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems*, 21(1):64–93, January 2003.
9. U. Keller, R. Lara, A. Polleres, I. Toma, M. Kifer, and Dieter Fensel. Wsmo web service discovery. http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112, 2004.
10. I. Elgedawy, Z. Tari, and M. Winikoff. Functional context matching for web services. Technical Report TR-04-3, RMIT University, Australia, 2004.