Negotiating Robustness in Semantic Web Service Composition

Freddy Lécué, Usman Wajid, Nikolay Mehandjiev
The University of Manchester
Booth Street East, Manchester, UK
{(firstname.lastname)@manchester.ac.uk}

Abstract—Automating Web service composition is a challenging area of Service Oriented Computing, requiring progress on a number of issues concerning the automation of control flow, the alignment of data flow, verification, execution monitoring, etc. In this paper, we focus on aligning data flow in semantic web service composition to ensure the robustness when executing the composed service by preventing any cases when the wrong type of data is passed on from one service to the next. The proposed solution is unique in that it ensures the robustness of data flow when automatically composing web services through the use of agent-based negotiation between web service providers. Starting from the semantic specification of a non-robust composition, we adapt the mechanism of Concept Abduction to compute the set of additional semantic descriptions necessary to ensure robustness of the semantic connections within the composed service. An agent-based negotiation is then used to solicit the additional semantic definitions which are required for achieving robust web service compositions.

Keywords—Service Composition, Web Service, Semantic Web, Description Logic, Agent Negotiation.

I. INTRODUCTION

The Semantic Web extends the World Wide Web structures by giving semantic descriptions to web resources, be it web pages or web services [1]. Rich description languages based on Description Logics DLs [2], such as the Web Ontology Language (OWL) [3], are used to provide semantic annotations. A number of these languages are specialised for describing web services, including OWL-S [4], WSMO [5] or SA-WSDL [6]. We can thus define

Semantic web services [7] as web services that have been enhanced with formal semantic descriptions.

These descriptions allow automatic reasoning about the functionalities of Semantic Web Services. This reasoning can support the automated discovery, selection and composition of different services to accomplish the goals specified by service consumers.

A composition of existing web services is often used to overcome the problem where no single service can satisfy the goal specified by the service consumer. Starting from an initial set of available services, web service composition aims at selecting and inter-connecting Web services provided by different partners in order to achieve a particular goal.

In doing this, we need to consider many issues related to control flow, data flow, verification, execution monitoring, or recovery actions (e.g., compensation). In this work, we study robustness of data flow in semantic Web service composition, i.e., how to ensure seamless data integration in composition, avoiding cases where the wrong type of data can be passed from one service to another. To this end we focus on data which is exchanged through output and input parameters of web services. Even if some methods have approached this issue by discovering potential semantic connections between web services [8], no solution addresses the issue of robust data flow within a context of automated composition of web services. Indeed, once semantic connections between the output parameter of one service and an input parameter of another (also known as semantic links [9]) are elaborated during the composition process [9], [10], some of them can cause failure during the execution of the composed service.

For instance some input parameters of services may require data which is narrower in range than the data provided by the output parameters of other services. Thus, the compositions that are realized by semantic matchmaking are meaningful, but rarely robust and executable. This is mainly caused by heterogeneity of Web service descriptions, which affects the data integration within Web service compositions.

In this work, semantic links are considered mainly for i) guaranteeing the semantic connection and interaction between services, and ii) detecting areas of non-robustness in the data flow of any web service composition. From this we can compute the DL-based description of the data restrictions which can be added to the specifications of the service outputs to make the whole composition robust. Then we elaborate this approach into an agent-based solution, which allows the negotiation of these additional restrictions within the different web service providers, to ensure the implementation of the robust version of the web service composition. We illustrate the operation of the proposed approach using an extended example running through the paper.

The remainder of this paper is organised as follows. In the next section we review semantic links based web service composition, as well as the robustness characteristic. Section 3 presents an approach to compute a DL-based description of restrictions which can be added to turn a non-robust semantic link into a robust one. Section 4 presents the protocol used by agents to negotiate the acceptance of these additional restrictions. Section 5 discusses the novelty of the proposed approach compared to related work. Finally Section 6 draws some conclusions and possible future directions.
II. WEB SERVICE COMPOSITION

First of all, we present the concept of semantic links. Then we restate the definition of their robustness and finally we describe semantic-link-based web service composition.

A. Semantic Links

In the semantic web, input and output parameters of web services are specified using concepts from a common ontology\(^1\) or Terminology \(T\) (an example of such is given in Figure 3), where the OWL-S profile [4], WSMO [5] or SA-WSDL [6] can be used to describe them (through semantic annotations).

At functional level web service composition involves retrieving some semantic links between output parameters \(\text{Out}_{s_i} \in T\) of services \(s_i\) and input parameters \(\text{In}_{s_j} \in T\) of other services \(s_j\). Such a semantic link \([9]\) \(s_{i,j}\) (Figure 1) between two functional parameters of \(s_i\) and \(s_j\) is formalized as \((s_i, \text{Sim}_T(\text{Out}_{s_i}, \text{In}_{s_j})), s_j\)). Thereby \(s_i\) and \(s_j\) are partially linked according to a matching function \(\text{Sim}_T\).

The latter function expresses which matching type is employed to chain services. The range of \(\text{Sim}_T\) is reduced to the four well known matching type introduced by [11] and the extra type Intersection [12]:

- **Exact** If the output parameter \(\text{Out}_{s_i}\) of \(s_i\) and the input parameter \(\text{In}_{s_j}\) of \(s_j\) are equivalent; formally, \(T \models \text{Out}_{s_i} \equiv \text{In}_{s_j}\).

- **PlugIn** If \(\text{Out}_{s_i}\) is sub-concept of (i.e., more specific than) \(\text{In}_{s_j}\); formally, \(T \models \text{Out}_{s_i} \sqsubseteq \text{In}_{s_j}\).

- **Subsume** If \(\text{Out}_{s_i}\) is super-concept of (i.e., more general than) \(\text{In}_{s_j}\); formally, \(T \models \text{In}_{s_j} \sqsubseteq \text{Out}_{s_i}\).

- **Intersection** If the intersection of \(\text{Out}_{s_i}\) and \(\text{In}_{s_j}\) is satisfiable; formally, \(T \not\models \text{Out}_{s_i} \sqcap \text{In}_{s_j} \sqsubseteq \bot\).

- **Disjoint** Otherwise \(\text{Out}_{s_i}\) and \(\text{In}_{s_j}\) are incompatible i.e., \(T \models \text{Out}_{s_i} \sqcap \text{In}_{s_j} \sqsubseteq \bot\).

A semantic link valued by the Intersection matching type requires a comparable refinement. In this direction, [13] defined a robust semantic link and their composition.

\(^1\)Distributed ontologies are not considered here but are largely independent of the problem addressed in this work.

Example: Step 1. (Semantic Link & Subsume Matching Type)

Suppose \(s_1\) and \(s_2\) are two services such that the output parameter \(\text{NetworkConnection}\) of \(s_1\) is (semantic) linked to the input parameter \(\text{SlowNetworkConnection}\) of \(s_2\) (\(s_{1,2}\) in Figure 2). According to the example ontology in Figure 3, this semantic link is valued by a Subsume matching type since \(\text{NetworkConnection} \sqsupseteq \text{SlowNetworkConnection}\).

It is obvious that such a semantic link should not be directly applied in a service composition since the \(\text{NetworkConnection}\) is not specific enough to be used by the input parameter \(\text{SlowNetworkConnection}\), which may cause data-based exception during execution. Indeed the output parameter \(\text{NetworkConnection}\) requires further restrictions to ensure a data-robust composition of \(s_1\) and \(s_2\).

Figure 3. Sample of an ACL Terminology \(T\).

B. Robust Semantic Link

The matching function \(\text{Sim}_T\) enables us to determine the degree of semantic compatibility among independently defined web service descriptions at design time. This can range from the strongly compatible Exact through PlugIn, Subsume and Intersection to the strongly incompatible Disjoint. However, as emphasized by [13], the matching types Intersection and Subsume need some refinements to be usable for semantic-links-based web service composition.
iff the matching type between Out_s_i and In_s_j is either Exact or PlugIn.

**Property 1. (Robust Web Service Composition)**
A composition is robust iff all its semantic links are robust.

A possible way to replace a link \( \langle s_i, s_j, T \rangle \) valued by an Intersection or Subsume matching type in its robust form consists in computing the information (as DL-based description) contained in the input parameter In_s_j and not in the output parameter Out_s_i.

This information is then used as an additional restriction on the Out_s_i data type when a suitable web service is procured. We say that addition this latter restriction "transforms" the non-robust semantic link in its robust form.

To do this, we apply initial ideas of [8], which adapt a non standard inference matching type i.e., the Abduction operation [14], [15] (Definition 2) for comparing \( \mathcal{ALN} \) DL-based descriptions. Alternatively (depending on the DL expressivity), different inference matching types such as Difference could be employed (Section V-A).

**Definition 2. (Concept Abduction)**
Let \( \mathcal{L} \) be a DL, \( C, D \) two concepts in \( \mathcal{L} \), and \( T \) be a set of axioms in \( \mathcal{L} \). A Concept Abduction Problem (CAP), denoted as \( \langle \mathcal{L}, C, D, T \rangle \) aims at finding Extra Description, as a concept \( H \in \mathcal{L} \) such that \( T \models C \cap H \subseteq D \).

According to Definition 2, a compact representation of "difference" \( H \) between DL-based descriptions Out_s_i and In_s_j of a semantic link \( s_i, s_j \) can be computed.

Such a description \( H \) is formally defined by \( T \models Out_s_j \cap H \subseteq In_s_i \) as a solution of the Concept Abduction problem \( \langle \mathcal{L}, Out_s_i, In_s_j, T \rangle \). In other words the Extra Description \( H \) refers to information required by In_s_j but not provided by Out_s_i to ensure a correct data flow between web services \( s_i \) and \( s_j \).

**Example: Step 2. (Robustness and Extra Description)**
Suppose the semantic link \( s_{1,2} \) in Example: Step 1 and Figure 2.

Such a link is not robust enough (Definition 1) to be applied in a robust composition since \( T \models NetworkConnection \sqcap SlowNetworkConnection \).

The additional restriction which has to be provided to the NetworkConnection if this output is to be used by the input parameter SlowNetworkConnection is referred by the Extra Description \( H \) of the Concept Abduction Problem \( \langle \mathcal{L}, NetworkConnection, SlowNetworkConnection, T \rangle \) i.e., \( \forall netSpeed \ Adsl1M \) (see Figure 3).

In other words, we can turn non-robust semantic links into robust ones by retrieving the Extra Description that changes an Intersection in a PlugIn matching type, and a Subsume by an Exact matching type.

**C. Semantic Link Composition Model**

Here, the process model of web service composition and its semantic links is specified by a directed graph which has the web service specifications as its nodes, and the semantic links (data dependencies) as its edges.

In addition some basic composition constructs such as sequence, conditional branching (i.e., OR-Branching), structured loops, concurrent threads (i.e., AND-Branching), and inter-thread synchronization can be specified, where their semantic is similar to the one they will be given within a statechart [16]. In the following we focus on data flow involved in any statechart-modeled semantic web service composition.

**Example: Step 3. (Process Model of a Semantic Link based Composition)**
Suppose \( s_{1,3} \leq s_{1,8} \) are six services extending Example: Step 1 and Figure 2 in a more complex composition. The process model and the data flow diagram of this composite service is illustrated in Figure 4. Such a composition consists in an OR-Branching and AND-Branching wherein nine semantic links (as data flow) are involved.

Given i) an approach to compute robust semantic links (Definition 2) and ii) the previous model of web service composition, we address the issue of automatically computing robust web service composition (Property 1).

**Example: Step 4. (Robustness of Web Service Composition)**
Suppose the composition in Example: Step 3 and Figure 4. In addition to \( s_{1,2}, s_{1,4} \) (valued by a Subsume matching type) and \( s_{7,8} \) (valued by an Intersection matching type) are two other non-robust semantic links. In this respect, the issue that we address is how to define the overall Extra Description that is required for robust composition?

**III. WHAT IS MISSING IN NON ROBUST COMPOSITIONS?**

Definition 2 provides the mechanism to compute the DL-based additional restriction which can be added to each non-robust semantic link to ensure its robustness. This description is useful for achieving robustness at local level (i.e., computing description of each non-robust link in an independently way and then requesting it). Therefore such a description can be provided separately by end-users, other services or any third party, depending on the application we want to automate.

Before automating the process that provides description for non-robust compositions, we focus on an overall method that gathers the different Extra Description in a more specific (in terms of Subsumption "\( \sqsubseteq \)" relationship) description, hence a more specific view of missing description in robust web service composition.

**A. Most Specific Description**

Suppose a non-robust composition of services wherein \( n \) semantic links are not robust enough. Therefore some Extra Descriptions \( H_{1 \leq i \leq n} \) are required to change non robust semantic links into their robust forms. Here we suggest to group them in a more specific form \( \mathcal{H} \) (Definition 3) rather than using the local description to achieve local robustness. In this definition
\langle s_i, \text{Sim}_T(\text{Out}_{s_i}, \text{In}_{s_j}), s_j \rangle \text{ refers to a semantic link in the composition model.}

**Definition 3. (Most Specific Description)**

The Most Specific Description \( H \), which is required in a non-robust composition is defined by the lower bound of its Extra Description. Formally \( H \) is defined as follows:

\[
H := \inf \{ H \in \langle \mathcal{L}, \text{Out}_{s_i}, \text{In}_{s_j}, T \rangle \} \setminus \{ T \}
\]

wherein \( \langle s_i, \text{Sim}_T(\text{Out}_{s_i}, \text{In}_{s_j}), s_j \rangle \) are non-robust semantic links and \( T \neq \text{Out}_{s_i} \cap \text{In}_{s_j} \sqsubseteq \bot \).

The Most Specific Description \( H \) of a Web service composition gathers different descriptions from the set \( \langle \mathcal{L}, \text{Out}_{s_i}, \text{In}_{s_j}, T \rangle \) where \( s_i \) and \( s_j \) are chained by a non-robust semantic link. Since Definition 3 computes the Most Specific Description, such a DL-based description should be used by a finite set of not robust semantic links to change them in their robust forms.

During computation, the relationship of each non-robust link to one of the tuple elements within the Most Specific Description \( H \) is stored separately, to allow the transformation into robust form. The descriptions used to perform these changes will be the Most Specific Descriptions of the Extra Description.

\( H \) does not only explain why the composition process failed but also gives a solution of the robustness problem of semantic links hence a way to reach robustness in semantic Web service composition.

**Example: Step 5. (Most Specific Description)**

Let us consider the composition depicted in Figure 4. Such a composition exposes a Web service composition wherein \( sl_{1,2}, sl_{1,4} \) and \( sl_{7,8} \) are not robust. Some Extra Description are required in order to obtain a composition of robust semantic links. The computation of \( H \) gives directions to solve local robustness of semantic links. According to Definition 3, the Most Specific Description \( H \) is constituted of conjunction of three Concept Abduction Problem in DL and their respective solutions:

1) \( \langle \mathcal{L}, \text{NetworkConnection}, \text{SlowNetworkConnection}, T \rangle \) to change \( sl_{1,2} \) by a semantic link valued by an Exact matching type i.e., \( \forall \text{netSpeed}.\text{AdslM}ax \)

2) \( \langle \mathcal{L}, \text{NetworkConnection}, \text{FastNetworkConnection}, T \rangle \) to change \( sl_{1,4} \) by a semantic link valued by an Exact matching type i.e., \( \forall \text{netSpeed}.\text{AdslM}ax \)

3) \( \langle \mathcal{L}, \text{VoIPId}, \text{IP Address}, T \rangle \) to replace \( sl_{7,8} \) by a semantic link valued by a PlugIn matching type i.e., \( \forall \text{protocol}.\text{IP} \).

Since other semantic links are robust, and \( \forall \text{netSpeed}.\text{AdslM}ax \sqsubseteq \forall \text{netSpeed}.\text{AdslM}ax \), \( H \) is defined by

\[
\{ \forall \text{netSpeed}.\text{AdslM}ax, \forall \text{protocol}.\text{IP} \}
\]

Each DL-based element of \( H \) is attached to their non-robust semantic links in order to solve robustness. For instance \( \forall \text{netSpeed}.\text{AdslM}ax \) will be attached to semantic links \( sl_{1,2} \) and \( sl_{1,4} \) whereas \( \forall \text{protocol}.\text{IP} \) will be attached to \( sl_{7,8} \).

In this work, the Most Specific Description \( H \) is computed independently (by means of a DL reasoner) of the composition constructs involved in the composition. However this definition can be adapted by considering, for instance, different approaches to compute the Most Specific Description i.e., we can compute different levels of description on each OR-Branching and then consider all of them during the robustness evaluation. We can also imagine alternative approaches for compositions wherein some structured loops are involved.

During all manners of computation we need to preserve the link between each non-robust link such as \( sl_{1,2} \) and its tuple element in the Most Specific Description \( H \) (in this example \( \forall \text{netSpeed}.\text{AdslM}ax \)), to allow for the effective procurement of web services at the next stage.

**B. XML Encoding of Extra Description**

Since we aim at automatically providing robust composition of web services, we present a common interface (as an XML encoding) that any third party such as web services or agents can interact with in order to manipulate and reason on Extra Description and their most specific form. To this end we adapt the DIG interface specification\(^2\) [17] in order i) to model any Extra Description \( H_k, 1 \leq i \leq n \)

\(^2\)http://dig.cs.manchester.ac.uk/.
or $\mathcal{H}$ (Definition 3), and ii) to provide uniform access to DL reasoners, hence facilitating discovery or negotiation of the latter description by means of DL reasoning processes.

In more details we considered i) a subset of the DIG (DL Implementation Group) overall specification by focusing only on its Concept Language (actually any language no more expressive than $SHOIQD$ such as the $ALN$ DL language we use in this work), and more specially its XML Schema. In such a subset

- **Atoms** such as Concepts and ObjectProperty;
- **Concept Operators** such as AND ($\cap$), ATLEAST ($\geq n$), ATMOST ($\leq n$), ALL ($\forall$),

are considered for the XML definition of any Extra Description (as well as their most specific form).

**Example: Step 6. (XML Encoding of Extra Description)**

Suppose $\forall netSpeed.AdslMax$, $\forall protocol.IP$ is the Extra Description in Example: Step 5. Its XML encoding is defined as following:

```xml
<xmlversion="1.0" encoding="ISO-8859-1">
<REQUEST
xmlns=http://dl.kr.org/dig/2003/02/lang
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=http://dl.kr.org/dig/2003/02/lang
http://dl-www.man.ac.uk/dig/2003/02/dig.xsd
uri=urn:uuid:abcdefgh-1234-1234-12345689ab>
  <defconcept name="ED1"/>
  <equalc>
    <catom name="ED1"/>
    <all>
      <ratom name="netSpeed"/>
      <catom name="AdslMax"/>
    </all>
  </equalc>
  <defconcept name="ED2"/>
  <equalc>
    <catom name="ED2"/>
    <all>
      <ratom name="protocol"/>
      <catom name="IP"/>
    </all>
  </equalc>
</REQUEST>
```

where ED1 and ED2 are respectively the Extra Descriptions $\forall netSpeed.AdslMax$ and $\forall protocol.IP$.

Once the set of Extra Descriptions $\mathcal{H}$ and $\mathcal{H}$ (Definition 3) are computed through Concept Abduction, a process that automatically compiles them (for instance, from any service providers) is required. Here the Extra Descriptions will be negotiated with other agents representing web service providers. In other words the agents are then responsible of negotiating the Extra Description that the system needed to elaborate the final robust composition of the system services, hence satisfying the initial user request in an automatic way.

IV. NEGOTIATING ROBUSTNESS OF COMPOSITION

Software agents have a high degree of autonomy, so they can decide when and under what conditions an action should be performed. This coupled with their interaction capability makes them suitable for facilitating dynamic/automated composition of services and ensuring its robustness as well, which would otherwise require manual coordination among various entities.

Existing approaches for automated service composition [18], [19], [20] require the existence of a central directory of service specifications, which restricts the scope of automation to computing compatibility and efficiency of potential compositions. However, in these approaches a composition is typically based on a single-iteration approach and fails to involve service providers in finding missing additional services.

In this paper, we propose an agent-based solution to address the complex issues underlying automation of robust service composition. Our approach exploits the inherent interaction capability of agent technology, which allows agents representing service providers to play a more active role in the composition e.g., agents can exchange counter proposals and impose conditions over the use of services. Finally, a negotiation mechanism as a process of reaching on agreement (provision of desired information and reducing inconsistency) has been used to realize robust service composition.

Negotiation is a process in which interested parties resolve, dispute or agree up on course of action. Agent-based negotiations typically imply the use of a negotiation protocol to support coordinated action on a specific matter or topic. An Agent Interaction Protocol (AIP) describes a communication pattern as allowed sequence of messages between agents. In Section III we have discussed the specific type of information which represents the topic of negotiation. This is complemented by the details of the negotiation protocol, provided in the next section.

A. Protocol for Robust Composition

In agent-based research a variety of negotiation protocols are proposed that support varying perspective of agents as well as facilitate different types of negotiations e.g., [21], also formalised as official FIPA specification. The generic and task independent nature of these protocols means that they can be used in different domains. For instance, Contract-Net protocol and English Auction protocols have been employed for agent-based service compositions [22], [23]. However, a common aspect of these protocols is that they give initiator agents the control over interaction (or negotiation), whilst the role of participating agents is limited to providing the required information.

In contrast, in the proposed approach agents are given a more active role. They can negotiate using proposals and counter-proposals to reach on agreement over varying perspectives. To enable such a behavior, a negotiation protocol is proposed.

The proposed protocol is a customized form of FIPA Iterated Contract Net Interaction Protocol. It allows agents to employ a range of strategies in terms of

3http://dig.cs.manchester.ac.uk/schema.xsd.


5http://www.fipa.org/specs/fipa00030/SC00030H.html
generating proposals; evaluating proposals and offering counter proposals [6] until an agreement is reached or the negotiation is terminated. In the following, we assume an agent (here the Initiator agent) being responsible for negotiating the provision of the most specific Extra Description $\mathcal{H}$ with other agents, called Participant agents. The latter represent different service providers, therefore they would be able to satisfy different aspects of the semantic descriptions that can potentially resolve the robustness issue. The main role of the protocol is to support the negotiation about the provision of the description $\mathcal{H}$. The agent negotiation process is summarized in following steps (Figure 5):

1) The interaction protocol starts when the Initiator agent sends a Call-for-Proposals (CFP) to the Participant agent(s). The CFP contains the XML encoding (as shown in Example: Step 6) in its message contents.

2) A Participant agent decodes the XML encoding and consults its service providers regarding the required Most Specific Description.

3) If the Participant agent is willing and able to contribute towards the robustness issue then it will respond with Propose, otherwise it will send Refuse. In Propose, the message contents may contain the required XML encoding of a proposed Extra Description $\mathcal{H}_p1$, which is subsumed by the Most Specific Description $\mathcal{H}$. This may be accompanied by costs or other conditions that the Participant agent wants to attach to this proposal.

4) On receiving a proposal (or proposals from several agents), the Initiator agent may decide whether to accept a proposal (and hence send an Accept-Proposal response), or to iterate the process by issuing a revised CFP with a new required description $\mathcal{H}_p2$. The latter is subsumed by the original Most Specific Description $\mathcal{H}$ and specifies the elements of $\mathcal{H}$ which are not yet covered by the set of received proposals $\mathcal{H}_p1$ (see M3 in Section IV-B for further details).

5) The protocol ends when the Initiator agent sends Accept-Proposal to a set of agents, or when it does not issue a new CFP.

6) After receiving Accept-Proposal a Participant can send an optional acknowledgment to the Initiator agent.

The graphical notation (Figure 5) provides a high level specification of the protocol. However, more detail is often required for the design of agents that take part in the protocol. For example Initiator agent requests proposals from Participant agents by sending a call-for-proposal (CFP). However, the graphical notation do not stipulate neither the conditions for the use of a particular communicative act (CA) nor the procedure employed by agents to generate their responses.

To define the communicative acts along with the conditions for their use, for each CA we specify any pre-conditions required for its utterance, the type of responses required and the impacts of the utterance on the information store of the agent.

S1) Call-for-Proposal($A_x, A_y, \mathcal{H}$), by initiating a CFP the Initiator agents $A_x$ can invite proposal from $A_y$ agents in the system about the proposition of the required Most Specific Descriptions $\mathcal{H}$.

Precondition: To initiate a CFP the initiator agents $A_x$ must be responsible for a composition which is not robust, and must have a calculated the Most Specific Descriptions - see M1, Section IV-B.

Expected Response:

$$\text{Response}(\text{CFP}(A_x, A_y, \mathcal{H})) = \begin{cases} \text{Refuse}(A_y \rightarrow x, \mathcal{H}) \\ \text{Propose}(A_y \rightarrow x, \mathcal{H}_{p1}) \end{cases}$$

where $\text{CFP}(A_x, A_y, \mathcal{H})$ refers to the call-for-proposal concerning $\mathcal{H}$. Message($A_y \rightarrow x, \mathcal{H}$) means that agents $A_y$ sends Message with $\mathcal{H}$ to one specific agent $A_x$. $\mathcal{H}_{p1}$ represents the proposal by the participating agents to a particular Initiator agent.

Information Store: No update.

S2) Propose($A_y, A_x, \mathcal{H}_{p1}$), where $A_y$ refers to the set of Participant agents. $A_x \in A_x$ is the initiator of

![Figure 5. Negotiation Protocol for facilitating robust service composition.](image-url)
the CFP such that $H_{p_1}$ refers to the proposal which is generated in response to the $H$ in CFP.

**Precondition:** A call-for-proposal must be published before the use of this CA. The agents $A_y$ must have the information e.g., $H_{p_1} \subseteq H$ that may result in the robust composition.

**Expected Response:** The initiator of CFP i.e. $A_x$, have the following options:

$$\text{Response}(\text{Propose}(A_y, A_{x_1}, H_{p_1})) = \begin{cases} 
\text{Accept-Proposal}(A_{x-y_j}, H_{p_1}) \\
\text{Reject-Proposal}(A_{x-y_j}, H_{p_1}) \\
\text{CFP}(A_{x_1}, A_y, H_2)
\end{cases}$$

The response should be communicated to the participating agents e.g. $A_{y_j}$ where $A_{y_j} \in A_y$, using the appropriate CA.

**Information Store:** For each proposal $H_{p_1}$, a tuple $(A_{x_1}, A_{y_j}, H_{p_1})$ is inserted into $IS(A_y)$.

**S3) Refuse($A_y$, $A_{x_1}$, $H_r$), where $H_r$ refers to the conditions for refusing $H$ in the previously received CFP.**

**Precondition:** A call-for-proposal must be published before the use of this CA and this CA cannot be uttered after Propose($A_y$, $A_{x_1}$, $H_{p_1}$).

**Expected Response:** Not required

**Information Store:** No update.

**S4) Accept-Proposal($A_{x_1}$, $A_{y_j}$, $H_{p_1}$), by using this CA the Initiator agent $A_{x_1}$ conveys the acceptance of the proposal $H_{p_1}$ which was submitted by the Participant $A_{y_j} \in A_y$.**

**Precondition:** A proposal in the form Propose ($A_{y_j}$, $A_{x_1}$, $H_{p_1}$) must have been received by the agent $A_{x_1}$ before the utterance of this CA.

**Expected Response:** Not required

**Information Store:** A tuple $(A_{x_1}, A_{y_j}, H_{p_1})$ is inserted into $ISA_x$.

Finally we specify the optional communicative act in the proposed protocol.

**S5) Inform($A_{y_j}$, $A_{x_1}$, $H$), by using this CA the Participant agent $A_{y_j}$ acknowledge the use of information in $H_{p_1}$ by the Initiator agent $A_{x_1} \in A_x$.**

**Precondition:** $A_{y_j}$ can only acknowledge the use of information in $H_{p_1}$ after it receives Accept-Proposal($A_{x_1}$, $A_{y_j}$, $H_{p_1}$) from the agent $A_{x_1} \in A_x$.

**Expected Response:** Not required

**Information Store:** No update.

**B. Enabling Automated Negotiation**

The specification of communicative acts and the condition for their use in Section IV-A highlight the syntactical aspects of the protocol. However, the generic nature of these rules needs to be complemented with specific processing and decision-making mechanisms. In this respect, we recognize high level agent functionality that is necessary for enabling automated negotiations among software agents.

**M1: Need for Most Specific Descriptions:** A mechanism that enable agents to compute and realize the need for Most Specific Descriptions (Definition 3), which is required for robust composition. The non-availability of Most Specific Descriptions can acts as a trigger for automated agents-based negotiation process supported by the proposed protocol.

**M2: Proposal Formation:** A mechanism for participating agents to compute the required information and generate a proposal. The mechanism also ensures that the same information (in the current proposal) is not already presented to the initiator agent in the protocol.

**M3: Proposal Evaluation and Ranking:** This mechanism allows the initiator agent to evaluate the received proposals and compute the best course of action. At every round of proposals, the initiator agent will calculate the set of outstanding elements of $H$ and, unless this set is empty, will proceed to invite proposals covering these outstanding elements. Depending on the configuration of individual initiator agents, the proposal gathering may stop when all known service providers have been contacted or after a fixed number of iterations, even if there are still outstanding elements not covered by proposals. The initiator agent will then proceed to evaluate and rank proposals using any of the known solutions to the set partitioning problem (see [24]). Further details about formulating this problem are provided below.

**M4: Notification of Decision:** A mechanism for initiator agent to notify participant agents about the outcome of their proposals. This mechanism follows M3, as shown in Figure 6.

**M5: Acknowledgement:** This mechanism allows the participant agent to acknowledge the use of its (previously proposed) information by the initiator agent. In contractual terms this acknowledgement can serve as the receipt for the use of service.
Mechanism 3 above uses the well-known set partitioning problem as formulated in [24] (equation (SPP) on page 712):

\[
\min \{c \times x | Z \times x = e, x_j = 0 \text{ or } 1, \forall j \in N\} \tag{3}
\]

We seek to minimise the cost function \(c \times x\), where \(c\) is the cost vector, containing the costs for selecting each provider, and \(x\) is the vector of choices about which provider’s offer to accept. Each provider is offering to provide some elements of \(\mathcal{H}\), these are recorded as values of 0 or 1 in the matrix \(Z\). We should have each element of \(\mathcal{H}\) covered exactly once, so \(Z \times x = e\) where \(e\) is a vector containing “1” in every element. Thus we should remove from consideration those elements of \(\mathcal{H}\) for which we have no offers.

We have proposed the high level functionality of the mechanisms that are required for automated negotiations among software agents. In this respect, Figure 7 shows how these mechanisms feed into the different communicative actions that are defined in Section IV-A.

**Example: Step 7. (Negotiation of Robustness)**

Upon computing the Most Specific Description \(\mathcal{H}\) provided in (2), the initiator agent sends out a CFP specifying \(\mathcal{H}\) as a condition. The CFP is addressed to a number of service provider agents, which are referred to as “Participants” in the protocol. One of the participants can provide \(s_1\), but instead of the condition \(\forall \text{netSpeed.AdslMax}\) for \(s_{1,2}\), they can propose a new condition \(\forall \text{netSpeed.AdslSuperMax}\). The latter is subsumed by \(\forall \text{netSpeed.AdslMax}\), so it still leaves the semantic link \(s_{1,2}\) robust. So the description proposed is

\[
\mathcal{H}_{p1} : \forall \text{netSpeed.AdslSuperMax} \tag{5}
\]

The Initiator accepts this proposal and sends out \(\text{AcceptProposal}(A_{\text{Initiator}} \rightarrow \text{Participant}, \mathcal{H}_{p1})\). Note that the description \(\forall \text{protocol.IP}\) (which is required by \(s_{1,2}\) to be robust) could be provided by the latter participants or any other as well.

Alternatively, they can propose a new \(\mathcal{H}_{p2}\), which requires the speed to be \(\geq 12\text{mBytes}\), hoping another recipient will agree on this, or the first recipient will drop their speed requirements.

V. RELATED WORK

In this section we discuss the advantages of the proposed approach compared to existing research work in the areas of i) Concept Abduction , ii) Robustness in semantic web service composition and iii) Agent-based Negotiation for web service composition.

A. Concept Abduction

Besides Concept Abduction, the Concept Difference [25] can be applied to evaluate robustness in web service composition. The difference operation of [25] requires a subsumption relation between descriptions to be matched since it requires the extra and strict condition \(C \sqsubseteq D\). This strict condition makes Concept Difference impossible to use in case of semantic links valued by an Intersection matchmaking type. Therefore robustness cannot be resolved by using such an operator.

Alternatively [26] proposed a refinement of this definition by taking the syntactic minimum instead of a semantic maximum in [25]. Its main advantages are as follows: i) firstly it does not contain redundancies in its result and ii) secondly it is more readable by a human user.

However Concept Difference [25] and [26] are more accurate, but more time consuming reasoning than Concept Abduction since they perform an equivalence between two concept descriptions (\(T \models B \cap D \equiv C\) or \(B \cap D \equiv C \cap D\)) whereas the Concept Abduction computes a subsumption of concept descriptions (\(T \models B \cap D \subseteq C\)).

B. Robustness in Composition

An intuitive method is discussed in [8] to immediately retrieve the Extra Description. Such a solution can be employed and implemented in any composition approach. In
case of a non-robust semantic link, the Extra Description is exposed to a Web service discovery process which is in charge of retrieving relevant Web services. The latter services are able to provide the Extra Description as output parameters. The Extra Description can be reached by one or a conjunction of Web services, depending on the Extra Description and the discovery process. The main constraint of this method is related to the complexity of composition. The more non-robust semantic links in a Web service composition the more important the cardinality of Web services will be implied in the composition. Moreover each input parameter of these Web services has to be either known at run time or linked to an output parameter of another Web service through a robust semantic link.

Alternatively the set of Extra Descriptions is suggested to the end user in [27]. This user is then responsible of providing the Extra Description that the system needed to elaborate the final composition. The new information that end users will provide to the system is necessary to compute and elaborate a robust composition of web service, hence satisfying the initial user request. The suggested method has the advantage of relaxing constraints on the end user. However, in this paper we suggest an automated approach which does not require any end user support.

C. Agent-based Negotiation

Software agents are considered a promising approach to service composition, partly because agent negotiations provide an effective way of addressing the complexity of a range of issues associated with automated service compositions [22]. Existing agent-based service composition approaches often rely on the use of a third party, which acts as a mediator between service consumers and service providers [28], [29]. However, the proposed approach removes the need for such a third party since it is based on direct negotiation between concerned parties to produce desired results. The focus of our approach is on the negotiation issue of providing extra semantic descriptions for achieving semantic robustness, and on the design of a protocol to facilitate negotiation among various agents. Details of the tactics that agents can employ in their negotiation are not discussed here, the reader is instead referred to [30].

VI. Conclusion

In the domain of realistic web service composition, one of the most important practical and theoretic concerns is that of ensuring robustness of the resultant compositions. Our paper has proposed an automated approach to ensuring robustness during web service composition, which can compute the DL-based descriptions that a non-robust composition requires to be robust, and negotiate their provision with service provider agents.

Our approach is based on Concept Abduction and its adaptation to ensuring robust composition of web services. From this we have suggested to compute a general DL-based description which is required to convert our non-robust composition into a robust one.

Once the description is computed, we propose the use of agent-based negotiation approach to retrieve the elements of this description from a number of potential service providers, and thus achieve robust web service composition. Our approach goes beyond the prevalent one-shot procedure (sending request and collecting results) by allowing agents to play a more active role in the composition process.

The approach is based on an iterative agent negotiation protocol, which allows agents to engage in negotiation with multiple agents by means of exchanging proposals and counter proposals. In this respect, the protocol facilitates agreement on service composition while allowing agents to decide on their actions and resolve their differences which may hinder robust composition. We have illustrated the operation of the proposed approach using an extended example running through the paper.

The main direction for future work is to consider robustness in more expressive composition of web services. In addition the issues involved in coupling semantic and syntactic data flow require further investigation.

Acknowledgments

This work is conducted within the European Commission VII Framework IP Project Soa4All (Service Oriented Architectures for All) (http://www.soa4all.eu/), Contract No. IST-215219.

References


