Abstract

Most of the work on automated Web service composition has focused so far on composition of stateful Web services. This level of composition so called “Process Level” considers Web services with their internal and complex behaviours. At process level formal models such as State Transition Systems (STS from now) or Interface Automata are the most appropriate models to represent the internal behaviour of stateful Web services. However such models focus only on semantics of their behaviours and unfortunately not on semantics of actions and their parameters. In this paper, we suggest to extending the STS model in the latter direction w.r.t the semantic Web. This semantic enhancement of STS so called \( S^2TS \) will enable to model semantics of internal behaviours and semantics of their actions together with their input and output parameters. Secondly, we will focus on automated generation of data flow (or the process to perform automated assignments between services of a composition). Thus we do not restrict to assignments of exact parameters (which is practically never used in industrial scenario) but extend assignments to semantically close parameters (e.g., through a subsumption matching). Our system is implemented and interacting with Web services dedicated on Telecom scenarios. The preliminary evaluation results showed high efficiency and effectiveness of the proposed approach.

1 Introduction

The semantic Web i.e., the Web of meaning is considered as the new vision and extension of the current Web that tries to give semantic to the Web resources. In such a new vision of the Web, Web services are semantically enhanced using rich description languages through Description Logics [3] such as the Web Ontology Language (OWL) [23]. Semantic Web services are then Web services that have been augmented with formal semantic descriptions where OWL-S [2], WSMO [6] or SA-WSDL [22] can be used to describe them. Therefore these descriptions can be used for reasoning and automating their use such as automated discovery, selection and composition. In particular Web service composition has triggered a considerable number of research efforts in the joint area of Semantic Web and Web service. Starting from an initial set of services, Web service composition aims at selecting and inter-connecting services provided by different partners according to a goal to achieve. Web service composition enhanced by semantic technologies is currently one of the most addressed issues in the Service Oriented Computing.

Most of the work in semantic Web services composition has focused on two main levels of composition: functional [17, 11] and process [4, 20] levels (respectively FLC and PLC). The former level considers Web services as “atomic” components described in terms of their IOPEs (inputs, outputs, preconditions, effects) and executed in a simple request-response step. The latter level supposes services as stateful processes with an interaction protocol (i.e., behaviour) involving different sequential, conditional, and iterative steps. In such a level of description stateful services expose their internal behaviours, and service composition aims at considering all distinct actions involved in Web services. Previous work in this regard has considered finite state machine (i.e., State Transition System STS [20]) or Interface Automata [1] to model the internal and independent protocol of Web services [19, 15]. Even if the latter models are appropriate to represent semantics of stateful Web services, some information is still missing to describe semantic enhanced Web services. In particular semantics on actions and their parameters involved in the whole protocol of Web services cannot be modelled. Indeed the semantics of actions is restricted to its feature of input, output or internal action. Therefore it is not possible to apply any kind of reasoning on these syntactic descriptions. In this work, we introduce the semantic enhanced STS (\( S^2TS \) from now) as a new way to model the internal behaviour of stateful services together with its data flow (well known as the assignment

\*Foundation Project: Supported by European Commission VII Framework IP Project Soa4All.
process between Web services) at semantic level. STSs and $S^2$TSs as well could be used to model Web services and composition of services as well. The extension proposal has roots in semantic Web since the annotation of actions and their parameters are key to obtain real semantics of Web services at process level. Secondly, we suggest to study the automated generation of data flow in semantic Web service composition. By considering the semantic-enhanced model $S^2$TS, assignments of not only exact parameters but also semantically close parameters is conceivable. Unlike current models that fail to automatically and semantically discover data exchanged between services through their protocols, the suggested model eases the automated generation of data flow hence the Web service composition too.

The rest of the paper is organized as follows. Section 2 presents the scenario wherein we integrated the suggested approach. In Section 3 we introduce $S^2$TS i.e., a model to represent stateful services together with semantics of their actions and respective parameters. Section 4 sketches advantages of $S^2$TSs to perform automated generation of data flow in service composition. Section 5 presents a prototype implementation of the proposed approach, as well as an experimental evaluation. Finally, Section 6 discusses related work, and Section 7 draws some conclusions.

2 An Industrial Scenario in Use

In the Internet domain commercial offers of Telecom operators are used to be composed by a set of more technical offers. So far the end users can only choose pre-existing commercial offers from the set of available offers. Such an approach is far from convenient for end users since the pre-existing commercial offers cannot still satisfy end user constraints and preferences. The main motivation is to give to end users the possibility to create their own commercial offers according to their real needs, without any kind of assistance: automatic generation of personalized Internet offers. The result will be a complete customized offer wherein the end user selects the offer(s) she wants to subscribe. For instance the offers may be as follows, ADSL eligibility, LiveBox, Voice over IP, Address Book, IPTV (TV over IP). Each offer is interface by a semantic Web service (e.g., IPTVService interfaces the IPTV offer), facilitating flexible and scalable applications through the loosely coupled feature of Web services. Moreover each Web service is also represented by its own internal protocol (e.g., the protocol of LiveBox service is drawn in Figure 2). In addition all services are described (in terms of input, output parameters and actions) by means of an ontology (Figure 1, exposed a portion of the $ALN$ Terminological Box of the ontology $T$).

The ultimate goal is to provide a correct composition of the latter Web services w.r.t not only i) the semantic con-

\[\begin{align*}
\text{NetworkConnection} &\equiv \forall \text{netSpeed.Speed} \\
\text{NoNetworkConnection} &\equiv \text{NetworkConnection} \land \forall \text{netSpeed.NoAdsl} \\
\text{SlowNetworkConnection} &\equiv \text{NetworkConnection} \land \forall \text{netSpeed.AdslMax} \\
\text{FastNetworkConnection} &\equiv \text{NetworkConnection} \land \forall \text{netSpeed.AdslMax} \\
\text{Speed} &\equiv \forall \leq 0 \text{mBytes} \\
\text{NoAdsl} &\equiv \forall \leq 0 \text{mBytes} \\
\text{AdslMax} &\equiv \forall \geq 1 \text{mBytes} \\
\text{AdslMax} &\equiv \forall \geq 8 \text{mBytes} \\
\text{IPAddress} &\equiv \forall \text{protocol.IP} \\
\text{VoIPId} &\equiv \forall \text{network.FTLocal} \\
\text{VideoDecoder} &\equiv \forall \text{decrypt.Video} \\
\text{ZipCode} &\equiv \forall \text{T, EMail} \\
\text{OutAction} &\equiv \forall \text{test, InterAction} \\
\text{orderAction} &\equiv \forall \text{PhonNum} \\
\text{findOfferAction} &\equiv \forall \text{1 findOfferAction} \\
\end{align*}\]

Figure 1. Part of the domain $ALN$ ontology.
3Semantics and Behaviour

Here we suggest a way to model internal choreography and non determinism of stateful Web services but also semantics of their actions and messages. Unlike most of works in Web service area which model only behaviour of Web services by means of e.g., STS [15, 19] (well known as a simple but powerful formalism to capture a large class of stateful services), Finite state automata [4, 8], Mealy machine [10] or Interface automata [7], we suggest to augment the STS model by semantically annotating actions together with their parameters. Such annotations will be used to improve semantics of the existing model and more specifically useful to make possible automated generation of data flow inside the internal protocol of a Web service as well as between different Web services in case of a composition.

3.1 S^2TS i.e., the Semantic Enhanced STS

Since the model introduced in this paper is a semantic extension of STSs (S^2TS from now), S^2TS shares the main features of STSs. Therefore a S^2TS defines a dynamic system that can be in several possible states (some of them are marked as final states or initial states) and can evolve to new states as a result of performing some actions. In other words STSs as well as S^2TSs model the behaviour of a Web service as a sequence of transitions wherein some deterministic choices are done. Moreover the non determinism criterion of Web services can be still represented by a STS and its semantic extension S^2TS as well. In the same direction as the STSs [19], the transitions of S^2TSs refers to actions i.e., internal, input and output actions. Each state represents the history of executed actions. In the following process algebra formalism [9] is used to model internal behaviour of Web services e.g., a receive and send action are respectively denoted by ?a and !a. Roughly speaking input actions represent the perception of messages i.e., an action ?a is received by the service and provides a set of output parameters required by other actions of the service. Output actions represent messages sent to external services i.e., an action !a is sent by the service itself and requires a set of input parameters to be sent. The action η is used to represent internal evolutions that are not visible to external services i.e., internal actions consumed some input parameters and produce some output parameters which are not directly available for external services. From an external view the state of the system can evolve without producing any output, and independently from the reception of inputs. A transition relation describes how the state can evolve on the basis of input, output or internal actions.

Example 1. (STS: a syntactic description of services)
Suppose the STS model to represent semantics of stateful services. Let LiveBox service be the Web service described in the previous section and its internal behaviour modelled by the STS in Figure 2. Such an automaton specifies LiveBox as a service which receives an input action i.e., an !order with some output parameters, then checks the availability book_available of an item through an internal action, then computes the !invoice in order to send it to another service (e.g., a mediator that orchestrates messages received) through an output action, and finally waits an acknowledgement ?ack or ?nack. The STS description of services enables to reason about semantics of its behaviour but not about semantics of its actions. As previously said in introduction the only semantics of actions is about their feature of input, output or internal.

![Figure 2. Internal Protocol of LiveBox.](image)

According to the previous example STS suffers from a lack of semantics, especially by considering their actions an parameters. Here we suggest a model wherein each evolution of a stateful and semantic Web service will be considered as a state transition labelled with actions and its input and/or output parameters. Actions and their parameters are semantically described by concepts in the TBox T of a domain ontology T.

Definition 1. (Semantic-enhanced Transition System S^2TS)
The semantic enhanced behaviour of a service i.e., a S^2TS is described as a 7-tuple (Q, Q_0, I, O, δ, F, T), where Q is a finite, non empty set of control states, with Q_0 ⊆ Q the initial states and F ⊆ Q the final states. T refers to the TBox T of a domain ontology which describes knowledge in a particular domain. I ⊆ T and O ⊆ T are input and output alphabets (i.e., finite set of Input/Output actions). δ ⊆ Q × ((I × P(T)) ∪ (O × P(T)) ∪ {η} × P(T) × P(T)) × Q is the transition function of the S^2TS from a state to another state for either a pair in I × P(T), either a pair in O × P(T) or a triple in {η} × P(T) × P(T).

According to Definition 1 the semantic enhancement is mainly focused on input, output and internal actions in or-

---

1In the following P(T) refers to the powerset of the concepts specified in the ontology T. |P(T)| = 2^|T|.
order to disambiguate semantics of actions involved in a behaviour of a Web service. Such actions are described at semantic level through a TBox $T$ of a domain ontology $T$. In the same way each output parameters of input actions, input parameters of output actions, input and output parameters of internal actions $\eta$ referred to concepts in $T$. State transitions of such an automaton are then defined by $\delta \subseteq Q \times ((I \times P(T)) \cup (O \times P(T)) \cup \{\{\} \times P(T) \times P(T)) \times Q$. An input action $I \times P(T)$ contains semantics of the action $I$ and its output parameters $P(T)$. An output action $O \times P(T)$ contains semantics of the action $O$ and its input parameters $P(T)$. An internal action $\{\} \times P(T) \times P(T)$ contains semantics of the action $\eta$ and both input $P(T)$ and output parameters $P(T)$. Input and output actions are both containing the distinguished element $\epsilon$ i.e., the empty action that provides neither output parameters (e.g., $\{\epsilon, \emptyset\}$) and consumes nor input parameters (e.g., $(\epsilon, \emptyset)$). Such an action is not internal since it is visible from external Web services. This action is significant to link two states of two distinct Web services e.g., a final state and initial state of two Web services.

Remark 1. (STS vs. $S^2$TS)

The $S^2$TS transcription of the STS of the LiveBox service (Figure 2) is illustrated in Figure 4. It is obvious that the semantics of its behaviour is unchanged since the STS skeleton is reused by the $S^2$TS model. However the $S^2$TS model of this service provides more semantics on its actions and its parameters. In a nutshell each action and its parameters are semantically augmented by annotating them with concepts in $T$. For instance the input action $\text{order}$ of LiveBox and its output parameters are annotated with their semantic definition in $T$. In this direction the syntactic action $\text{order}$ is annotated by the concept $\text{orderAction}$ in $T$ (Figure 1). In the same way parameters $\text{phoneNum}$, $\text{anAddress}$ and $\text{aDecoder}$ are annotated by their semantic definition in $T$ i.e., respectively $\text{PhoneNum}$, $\text{IP Address}$, and $\text{Decoder}$.

Adding semantics on actions and their parameters is the main contribution of the enhanced semantic transition system. Even if such the approach is straightforward, the difference with STS is really significant.

3.2 $S^2$TS and Reasoning

Automated assignment of parameters is still considered as an open and major issue [2, 15] especially for automated Web service composition. Roughly speaking data flow (a.k.a. assignment of parameters) in Web service composition is responsible for connecting and assigning parameters of distinct actions depending on their semantics. In such a case of assignment some output parameters of internal or input actions require to be used by some actions of other Web services to perform some new internal or output actions. Since $S^2$TS enables to model more semantics on actions and its parameters than current formalisms, it becomes conceivable to reason about these actions and their parameters. In particular it is then possible to perform automated generation of data flow in a composition of stateful Web services. To this end we assume a common ontology is used to annotate all actions as well the parameters of services.

3.3 Semantic Web Service Standards

As suggested by [20] state transition systems can be encoded with the standard proposal BPEL4WS to describe composition of services. In this way STS as well as its industrial encoding BPEL4WS are powerful enough to describe semantics of behaviours. However, as previously said, semantics on actions cannot be supported by BPEL4WS. Even if [18] introduced BPEL4SWS to overcome the latter issue, they do not address yet the semantic annotation of actions and their parameters. In this direction OWL-S [2] through its process model and WSMO [6] through its orchestration and choreography interfaces seem the most appropriate to model semantics of services behaviour together with semantics of actions. In contrary SWSL-FOL through the Process Specification Language (PSL) [12] is suitable for specifying process ontologies but not for semantics of actions and their parameters. SA-WSDL [22] is mainly based on a semantic extension of WSDL and then does provide semantics description of parameters but no description of the internal behaviours of stateful Web services. In this work we adapted and extended BPEL4SWS to encode $S^2$TS.

4 Service Composition & Data Flow

Since data flow is mainly required in composite Web services, we first explain the model we perform to achieve this composition and then to infer its data flow.

4.1 Service Composition Approach

As previously said Web service composition has mainly focused on two main levels of composition i.e., functional and process levels. Unlike most of methods that perform only one level of composition, in this work we suggest to achieve both. Thus we exploit FLC together with composition at process level due to their complementary [5]. First FLC is computed in order to obtain a partial order on Web services w.r.t the semantic dependencies retrieved between them. This partial order can model a composition of services by means of sequences and parallel branches of services e.g., composition of services described in Section 2.
The dependencies used to model the partial order of Web services are semantic links [13] between Web services parameters (i.e., output and input). Therefore FLC aims at identifying which external connections (by means of semantic links) should be required between two Web services involved in the composition. In such a level of abstraction Web services are considered as stateless services, without any kind of internal behaviour. Once FLC is performed, PLC is applied considering this partial order [13, 11]. Therefore PLC is not performed from scratch. In this next level, Web services are considered with their complex and internal behaviour. In our approach the main task of PLC consisted in the automated generation of data flow between Web services involved in the partial order. To this end we assume that behaviours of Web services are described according to the S2TS formalism in order to model the two levels of semantics i.e., semantics of behaviour and semantics of actions and their parameters. By describing the internal protocol of Web services according to S2TS we will be able to automatically infer semantic assignments between Web services involved in the FLC.

**Example 2. (Assignment in Web service composition)**

Suppose the motivating example in Section 2 such that the services are described according to Definition 1. Once FLC is achieved the partial order states that the TVsOverIP service required to be performed before the LiveBox service (semantic link sl1 in Figure 3). At functional level the output parameter VideoDecoder is used by the input parameter Decoder whereas at behaviour level the output parameter VideoDecoder of the TVsOverIP service (Figure 4) will be used by the action ?order of the LiveBox service (Figure 4). In other words there is an assignment of a parameter between two actions of two different services. However such an automated assignment of parameters is not straightforward without considering semantics on actions and their parameters.

**Figure 3. FLC of the Motivating Example.**

In the following we describe an approach for automated generation of data flow in web service composition.

### 4.2 Requirements for Data Flow

By assuming a partial order on Web services and a S2TS description of their behaviours, the main idea consists in expanding semantics along the internal protocol of each Web service, and then along this partial order. Such expansions are depending on i) the semantics of actions parameters, and ii) a new and simple semantic operator $\rightarrow_{T}$ [14] we automatically apply along protocols of Web services. This operator will be used to discover if two parameters of two independent actions require a process of semantic assignment. Since parameters of actions are semantically enhanced it becomes conceivable to infer requirements of assignments between some parameters of actions. More formally this operator so called T-semantic implication (Definition 2) requires two set of concepts in $\mathcal{P}(T)$ and returns a boolean true in case an assignment of parameters is required between some concepts of these two sets and false otherwise. In the same way we could define the T-semantic equivalence between two sets of parameters.

**Definition 2. (T-semantic implication)**

The set of concepts $F \subseteq \mathcal{P}(T)$ is a semantic implication of $E \subseteq \mathcal{P}(T)$ (i.e., $E \rightarrow_{T} F$) if and only if $\forall f \in F, \exists e \in E$ such that $f \sqsubseteq e$.

According to the previous definition a semantic implication between two sets of concepts in $\mathcal{P}(T)$ is found in case there is a semantic dependence (through the subsumption relation $\sqsubseteq$) between some concepts of the two sets. In other words at least two concepts in $T$ of the latter sets are related by subsumption i.e., an instance of the former concept is an instance of the latter concept since the latter concept subsumes the former. Such an operator is very significant for automated assignment of parameters since it enables to retrieve data flow along the composition according to a subsumption relation. Indeed it is obvious that assignments are not only performed between exact concepts but also between subsumed concepts. Roughly speaking Definition 2 enables to infer that some parameters shares the same instance, hence a semantic assignment.

**Example 3. (T-semantic implication)**

Suppose the TBox $T$ of the domain ontology $T$ in Figure 1. We define $E$ by the parameters of the input action ?order of LiveBox service: PhoneNumber, IPAddress, Decoder, and $F$ by the parameters of the internal action getDecoderAvailability of the LiveBox service: Decoder. According to Definition 2 it is obvious that $E \rightarrow_{T} F$ is true since Decoder$_E$ is equivalent to Decoder$_F$. In other words a semantic dependence is discovered between an input and internal action of the same Web service. An assignment of parameters between two actions of the same service is retrieved. Such an assignment is straightforward in a manual way but more difficult to automatically infer with a STS model devoid of any semantic-based actions.

By Definition 2 it is conceivable to retrieve that input parameter of some actions (e.g., output and internal actions)
are related to output parameters of other actions (e.g., input and internal actions) or even from actions of other Web services. In such a case two Web services require an assignment of their parameters in order to automate an end to end composition of Web services.

### 4.3 Semantic Consistency of Service Behaviour

The Definition 2 enables to define and infer automated assignments in Web service composition but also semantic consistency of Web service behaviours. By semantic consistency of behaviours we ensure that each action of a service has correct assignments of their input and output parameters. In other words each input parameter of an action involved in a service have to be provided by a previous action in the current behaviour. For instance internal actions requires that their input parameters are assigned by other previous parameters if such an action is semantically consistent with its actions and behaviours. This consistency holds by applying and checking $T$-logical implication between the union of output parameters of all previous actions (in the protocol) and the input parameters of each action (formally $\cup_{a'} \subset a$ of $\text{output}(a') \rightarrow_T \text{input}(a)$). In other words it is necessary that the former actions (actions that provide output parameters) are performed first, then the latter action (action that uses the latter parameters) follows in the service behaviour. We say that the former actions $a_f$ precede the latter actions $a_l$ i.e., $a_f \prec a_l$. The actions $a_f$ require to be executed before $a_l$ since $a_l$ require and consume some parameters provided by $a_f$.

**Example 4.** (Ordering of actions) According to the service LiveBox depicted in Figure 4, ?order $\prec$ getDecoderAvailability since the the action ?order is executed first in the behaviour of the LiveBox service.

**Definition 3.** (Semantic consistency of a Web service) A service is semantically consistent with its actions and behaviour iff

$$\cup_{a'} \subset a \text{ output}(a') \rightarrow_T \text{input}(a) \quad (1)$$

for all input parameters $\text{input}(a)$ of all actions $a \in \{a, !a\}$ such that $a' \in \{a, ?a\}$.

Roughly speaking, a service is semantically consistent with its behaviour if and only if the input parameters of all output and internal actions are semantically implied by (at worst) the union of all output parameters of previous actions (internal and input actions) in the internal protocol of the considered service. In all the other cases the behaviour of services is not semantically consistent, hence cannot be involved in the composition process. Since composition of independent behaviours is a real issue, it seems relevant to ensure semantic consistency between the whole set of the latter behaviours.

**Example 5.** (Consistency of Web service behaviours) Suppose the LiveBox in Figure 4. The behaviour of such a service is semantically consistent since all input parameters of output and internal actions are provided by previous internal or input actions. For instance input parameters...
4.4 Automated Data Flow in Composition

Here we study in more details the algorithm 1 to perform the automated and correct assignments of parameters in a composition process. Unlike [19] we study an automated method to perform generation of data flow in Web service composition by means on semantic annotations of actions together with their parameters. By considering more semantics on actions and their parameters, we do not restrict to assignments of exact parameters (which is practically never used in industrial scenario) neither to manual assignment (which is very time consuming) but extend assignments to semantically close parameters (e.g., subsumption matching). Therefore the work suggested by [19, 15] can be easily adapted and upgraded by means of our model. Indeed we consider more complex cases of assignment in a more semantic model i.e., $S^2$TS.

Algorithm 1 performs automated assignments from a FLC by means of two main steps i.e., the internal assignment a.k.a the semantic consistency of behaviours (line 4) and the external assignment (line 11). The presented algorithm requires that semantics of Web service behaviours together with semantics of their actions are available. This seems conceivable in case Web services are formalized as $S^2$TSs. The main idea of step 1 consists in resolving the internal assignment of parameters inside each Web service behaviour. In this direction step 1 of the algorithm aims at retrieving output parameters of internal and input actions that could be used by input parameters of internal and output actions. To this end we apply $T$-semantic implication (line 7) between the union of output parameters and the considered input parameters. In case a $T$-semantic implication is true, an assignment (line 10) is required between some output parameters and these input parameters of (distinct) actions involved in services. In other words we retrieved some input or internal actions that provide output parameters which are subsumed by or equivalent to input parameters of an output or internal action. The second step (line 4) of algorithm 7 is in charge of linking dependencies between Web services by means of semantic links retrieved by the FLC. This step aims at discovering which actions are semantically linked together by their parameters.

In complex cases of composition wherein an input parameter of an (internal or output) action can be assigned with more than one output parameter of an (internal or input) action, automated assignments is no more conceivable with algorithm 1. Therefore a list of correct output parameters is then returned to the end-user, or developer to perform assignments in a semi automated way. This step ensures that assignments are controlled by end-user or developer in case its automation is not possible. In the suggested approach all input parameters are assigned with at least one output parameter since i) consistency of Web service behaviours are ensured (step 1 of algorithm 1) and ii) semantic links retrieved during FLC is correct (step 2 of algorithm 1).

**Algorithm 1: Automated Data Flow in Composition.**

```plaintext
1 Input: A functional level composition of Web services, A Domain Ontology $T$.
2 Result: A composition $\pi$ of Web service with assignments.
3 begin
4   /** Internal Assignments of parameter for each service. */
5   foreach Web service $s_x$ involved in composition do
6       foreach internal or output action $a$ involved in $s_x$ do
7           if $\exists a' \in s_x$ such that
8              $\cup_a^a'$ output($a'$) $\rightarrow T$ input($a$) then
9              foreach input parameter input$_a$ of a do
10                 $\exists a' \prec a$ such that
11                    input$_a$ $\sqsubseteq$ output$_a'$; add Assign(output$_a'$, input$_a$) in $\pi$;
12       /** External Assignments between parameters of distinct services. */
13       foreach semantic link ($s_y$, Sim$_T$(Out$_s$$_y$, In$_s$$_x$), $s_x$) in FLC do
14           foreach input action $a$ in $s_x$ do
15              $\exists a' \in s_y$ such that output$_a$ $\sqsubseteq$ input$_a'$; add Assign(input$_a'$, output$_a$) in composition $\pi$;
16      /** Return the composition with correct assignments. */
17   return composition $\pi$;
8 end
```

It is obvious that consistency of Web service behaviours (step 1 of algorithm 1) and external assignments (step 2 of algorithm 1) are guaranteed in case the domain ontology contains sufficient knowledge about the domain under consideration. In the opposite case step 1 and step 2 of algorithm 1 cannot be automatically generated in a correct way. The more the domain ontology is complete the more correct is the assignments in the service composition.

The computational complexity of the automated generation of data flow in a composition is function of i) the number of subsumption relationship retrieved in the step 1 of the algorithm i.e., related to the complexity of Web services behaviour (e.g., number of actions) and ii) the number of semantic links retrieved by FLC. The number of required assignment in a composition is lower bounded by the number of...
of semantic links in FLC.

Example 6. (Assignment of Parameters in Composition) A semantic link between the AdslEligibility service and the Voice-OverIP service (semantic link sl1 in Figure 3) can be modelled in two ways i)\((AdslEligibility, Sim_1(FastNetworkConnection, NetworkConnection), VoiceOverIP)\) and also ii)\((AdslEligibility, Sim_1(SlowNetworkConnection, NetworkConnection), VoiceOverIP)\) since there is a semantic relation (subsumption) between two output parameters of AdslEligibility (i.e., FastNetworkConnection and SlowNetworkConnection) and an input parameter of VoiceOverIP (i.e., NetworkConnection). More formally there are two actions offer in the AdslEligibility service and one action request in the VoiceOverIP service that can assign their parameters. Indeed the parameters SlowNetworkConnection and FastNetworkConnection are subsumed by the parameter NetworkConnection of the input action request.

5 Validation

5.1 Implementation

Our technique of “Automated generation of data flow” is implemented in the SLM (Semantic Link Matrix) model [13] i.e., a FLC model. Our approach extends the latter model by performing the automated generation of data flow. Here we briefly overview the prototype architecture and discuss its extension to support the data flow generation approach. The prototype architecture (Figure 5) consists of four main modules, namely the Discovery and Selection module, a Semantic Reasoning component, the FLC module, together with a pool of semantic-based services (i.e., interfaces of commercial offers), the whole extended by the Data Flow Generation component. From a Web-based interface the end user describes its personalized commercial offer by means of a set of offers she wants to subscribe. The personalized offer is rendered in the so called service goal, described in terms of its input and output parameters. From this the Discovery and Selection module facilitates the advertisement and location of relevant Web services. It is implemented using a semantic extension of Java implementation of the Universal Description, Discovery and Integration (UDDI with its TModel), the Web Service Modelling Language (WSML) to describe semantic Web services together with their domain ontologies, and an extended version of BPEL4SWS (i.e., an encoding of \(S^2TS\)) to model complex behaviour of Web services at semantic level. Once relevant Web services are retrieved, the FLC (i.e., based-SLM model) module aims at retrieving a partial order of these services by means of the Semantic Reasoning component. Roughly speaking the SLM model aims at retrieving semantic links between Web services, hence predisposing the FLC result to the automated generation of data flow. Finally, the Data Flow Generation component aims at computing all data flow required by the composite Web services, and rendering the latter service in \(S^2TS\) and then in an extended BPEL4SWS document.

The Service Editor provides facilities for defining new goal compositions (i.e., goal services) and editing existing ones. Moreover the composite service can be edited through a visual interface, and translated into the BPEL4SWS document for subsequent analysis and processing by the service orchestrator. The orchestrator is responsible for scheduling, initiating, and monitoring the invocations to the tasks of the composite service during its execution, and for routing events and data items between these components.

5.2 Experimentation

The PC used for running the prototype system had the configuration of Intel(R) Core(TM)2 CPU, 1.86GHz with 512 RAM. The PC runs Linux-gnu (2.6.12-12mdk) and Java 2 Edition v1.5.0_11.

Our Data Flow Generation approach have been evaluated on three scenarios in use in France Telecom:

i) one in the Telecom domain (the extended version of the motivating scenario in Section 2) where the number of potential Web services \(#S^*_{Ws}\) is 35 and the TBox of the \(ALN\) Ontology consists of 305 defined concepts and 117 object properties;

ii) another in the E-Tourism domain where \(#S^*_{Ws}\) is 45 and the TBox of the \(ALN\) Ontology consists of 60 defined concepts and 19 object properties;

iii) and finally one in the E-HealthCare domain where \(#S^*_{Ws}\) is 12 and the TBox of the \(ALN\) Ontology consists of 105 defined concepts and 37 object properties.

The purpose of this experiment is to value the scalability of the Data Flow Generation (through its Internal and External Assignments), and compare it with the other components of the architecture depicted in Figure 5.

The whole execution of the architecture takes at worst 0.5 second. This is much faster than it would take for a user to look for services, and to interact with each of them.

From the table 1 and figure 6 (testing on complex scenarios with services from 5 to 100), we can see that Data Flow Generation is more time consuming than FLC. More particularly the internal assignment is greedier than the external assignment. This seems correct since internal assignment is more or less close to the complexity of Web services behaviours whereas the external assignment is simply related to the number of semantic links in the FLC.
atomic services wherein the latter services are simply linked by a semantic relation between their parameters. By considering a stateful Web service with its own internal behaviour, we focus on semantic link between the latter behaviours i.e., assignments between parameters of their actions. More precisely we study in more details the data flow i.e., data that are exchanged along actions of component services. In PLC AI planning [16] e.g., with HTN [21], under uncertainty [5] or as model checking [20] is used to perform composition. In such approaches composition is performed from a set of services described by their internal behaviour and a composition goal to achieve. Unfortunately automated generation of data flow in the composition is not studied and still an open issue in service composition. An another approach [4] performs a synthesis of behaviours rather than an orchestration of services. Therefore they retrieve a composition of available behaviours in order to satisfy a target behaviour. However data flow and its automated generation is not addressed in these works.

Given a composition goal, [19] automatically generate a knowledge level representation that “simply” declares what the composite service must know and how goal variables must be related with variables of component services. To this end, they must provide a further Knowledge Base so called the Knowledge base of the goal. In such a model goal variables and functions need to be previously determined hence making harder the automated generation of data flow. Even if [19, 15] suggest the use of typed functions and variables, these latter types are only used to perform exact matching without any kind of semantics. Thus they assume that the assignment process is performed between only exact parameters, hence a very restrictive assumption.

6 Related Work

Since Web services are stateful, service composition cannot be reduced to an AI planning problem [13, 21, 16] of...
The main motivation of the data flow modelling language [15] is: the specification of complex requirements for data manipulation. In this way they introduced a set of basic elements to make different format of data exchanged (assignments: fork, merge...). These different kinds of format are useful to model the data flow the end user want but not its automation. Indeed such a model requires the end-user knows exactly the behaviour she want. Unlike their model we suggest to automate data flow according to a semantic model $S^2TS$.

Many upcoming standards may be used to formalize assignments between Web services such as WSBPEL (i.e., assign) or OWL-S (i.e., by the class ValueOf with rdf properties atProcess and theParameter) since data flow in Web service composition is considered as essential. However at the best of our knowledge there is no model specifying data flow in an automated and semantic way.

7 Conclusion and future work

We studied stateful service composition and more specifically the automated generation of its data flow. This automation is essential to perform automated end to end Web service composition since data flow is often performed manually or in a semi automated way once the composition skeleton is computed. Such an issue is far from trivial since data flow of the whole composite service have to be inferred from a underspecified semantic description of services’ protocols. Most of work focused on models that represent behaviour of stateful service, or semantic of functional parameters, but unfortunately not both. Here we study in more details the internal behaviour of services together with the semantics of their actions. In this direction the semantic enhancement of STS i.e., $S^2TS$ aims at easing the automated and dynamic assignment (i.e., data flow) between services involved in the composition process. The only assumption is that composition at functional level is first achieved.

Since Web services can be modelled by the semantic enhancement of a STS, other kinds of reasoning can be considered as an interesting feature for $S^2TS$. For instance providers of Web services could perform automated reasoning to infer which actions can be replaced by (an) other(s) action(s). Another direction is concerning the extension of our approach with cyclic $S^2TS$s.

References


