Semi-Automated Service Composition using Visual Contracts

Muhammad Naeem
Department of Computer Science
University of Leicester, UK
mn105@mcs.le.ac.uk

Reiko Heckel
Department of Computer Science
University of Leicester, UK
reiko@mcs.le.ac.uk

Fernando Orejas
Department of Software Technology
Technical University of Catalonia, Barcelona
orejas@lsi.upc.edu

ABSTRACT
Services provide access to software components that can be discovered dynamically via the Internet. The increasing number of services a requesters may be able to use demand support for finding and selecting services. In particular, it is unrealistic to expect that a single service will satisfy complex requirements, so services will have to be combined to match clients’ requests.

In this paper, we propose a visual, interactive approach for the composition of services, in which we describe the requirements of a requester as a goal which is matched against multiple provider offers. After every match with an offer we decompose the goal into satisfied and remainder parts. We iterate the decomposition until the goal is satisfied or we run out of offers.

Finally, the individual offers are combined into a single combined offer and shown to the requester for feedback. Our approach is based on visual specifications of pre- and post conditions using the theory of graph transformations systems as a formal background.

1. INTRODUCTION
Service-oriented Architecture (SOA) [1] support dynamic discovery and binding based on matching requester’s requirements with providers offers. Both requirements and offers can be expressed as specifications of the (expected or given) semantics of a service’s operations in terms of their pre- and post conditions. At a technical level this is supported by semantic web technologies (e.g. OWL-S [2], WSML [3]), at modelling level visual contracts have been suggested to describe service semantics [4].

However, expecting to find a single service for each requirement is unrealistic. Often services need to be combined to satisfy the demands of clients. For example, let us consider a scenario, where a requester is looking to book a trip for attending a conference. The requester may be interested in flight and hotel reservation. Rather than using a single service, the requester may have to use two separate service providers.

In this paper we propose an interactive approach for service composition, where we assume that the requirements are expressed by a single goal stating pre- and post conditions. A variety of offers could contribute to the goal, each described by pre- and post conditions as well. We propose a notion of partial matching of offers with goal. After every partial match we compute the remaining requirements by decomposition of the original goal into the satisfied subgoal and its remainder. We iterate this process until the goal is achieved or we do not have any more offers.

As a result of this procedure we produce a combined offer which can be visualized and reviewed by the client. Our approach thus supports semi-automatic service composition.

The rest of the paper is organized as follows. Section 2 discusses related approaches. Section 3 introduces the graphical description of services before Section 4 discusses the need for partial matching of services. Section 5 presents our approach towards iterative based on the construction of the remainder rule and generation of combined offers from individual ones, and Section 6 discusses the main results and concludes the paper.

2. RELATED WORK
The challenge of describing dynamic discovery of web services has been the focus of a number of scientific publications. As already mentioned that it is unrealistic to expect a single offer to be satisfiable for every goal, offers have to be combined to satisfy the goal.

We believe, whenever multiple offers are going to be matched with a single goal then the following proposed requirements must be addressed:

- **Partial Match:** Does the approach support partial matching of an offer with a goal, or is full satisfaction of all requirements necessary?
- **Flexibility:** Does the approach allow to match offers in arbitrary order or does it follow a given control flow?
- **Completeness:** Is the approach decidable, i.e., does it provides a terminating procedure to find out if there are combinations of offers satisfying the goal?
In [7], authors have shown two step approach for semi automatic service composition. Firstly, they define the fixed part of process in the composition service by utilizing BPEL4WS technology and to pre-bind the corresponding service. Secondly, to dynamically and temporarily define the unfixed part which is determined by the diverse, uncertain and variable factors resulting in frequently service change. The proposed approach utilizes process ontology, BPEL4WS and OWL-S but it lacks to provide the concept of partial match and feedback to the user. In [8], authors have shown a flexible framework for semi-automatic web services composition based process ontology. In fact authors have improved their own work of [7], by considering non-function requirement of web services such as response time, usability, accessibility, integrity, reliability etc. again their approach based on complete match and do not provide feedback to the user.

In [9] authors proposed DAML-S process model which support AI planning techniques for automatic web service composition called SHOP2. Authors use logic to represent pre- and post conditions for the operations. They guarantee the termination based on the assumption of finite number of computations. While presenting only semantic approach their goal is to find and accumulate the independent processes which contribute towards a top level composite process. Our approach supports visual contracts based on the formal background of graph theory, so our approach could get the attraction of audience who are not only interested in semantic but also in visual representation of operations.

In [10], authors have proposed to integrate semantic web technology for automating customized, dynamic binding of web services together with interoperation through semantic translation based on BPEL4WS. While presenting the bottom-up approach, authors claimed for automatic binding and selection between multiple service partners based on user-defined preferences and constraints, and Integration of service partners with syntactically distinct but semantically translatable service descriptions. Their approach is not only flexible but also complete and provide feedback to user but they have not provided the partial matching concept between service partners.

In [11], authors have shown a comprehensive survey on composition of web services until 2005. Our approach differs from the above mentioned approaches, and these differences can be categorized as:

- Our approach is visual, goal and offers (both individual and combined) can be described using graphical pre- and post conditions.
- We do helpful partial match of a single goal with multiple individual offers.
- Our approach is decidable i.e. we have provided terminating condition for matching (Section 4).
- We provide combined offer at the end of service composition which can be reviewed by requester.

### 3. GRAPHICAL SERVICE DESCRIPTION

Services are dynamic in nature. They provide interfaces to either publish or request certain functionalities of a complex software system. The provider has to decide which of the operations in the application is suitable to be published as a service interface. We are using graph transformations [12, 13, 14] to specify service descriptions.
Graph transformations provide a modelling language where instance graph represents states and rules specify state changing operations. The class of admissible states is specified by a type graph [15, 16, 17]. Fig. 1 shows the type graph for the scenario discussed in Section 1. The type graph serves as an abstract representation using the notation of UML class diagram [18, 19, 20, 21]. Service descriptions and instance graphs are given over this type graph which serves as a fixed ontology for the service.

Next, we present a semi-formal account of the concepts of graph transformation used in the paper. For a formalization in terms of category theory see [22, 23]. A rule $p : L \rightarrow R$ is a pair of graphs where the left-hand side $L$ represents the pre-conditions and the right-hand side $R$ specifies the post conditions/effects. Applying the rule $p : L \rightarrow R$ to a given graph $G$ (representing the current state) means finding an occurrence of $L$ in $G$ and replacing it by a copy of $R$, leading to the derived graph $H$ [22, 23]. This is denoted by $G \xrightarrow{p} H$.

An example is shown in Fig. 2, with rule $res$ in the top of the picture and the graphs $G$ and $H$ at the bottom. The occurrence $o_L$ of $L$ and $G$ means that Client, CreditCard and Trip objects are present in the state, and connected as shown in the rule’s left-hand side. The transformation creates a new Flight object and reduces the remaining budget of the Trip objects are present in the state, and connected as shown in the top of the picture and the graphs $G$ and $H$ at the bottom. The transformation creates a new Flight object and reduces the remaining budget of the Trip object and replaces it by a copy of $R$, leading to the derived graph $H$ [22, 23]. This is denoted by $G \xrightarrow{p} H$.

Formally, there exists a transformation $G \xrightarrow{p o} H$ using rule $p : L \rightarrow R$ iff there is an occurrence $o : L \cup R \rightarrow G \cup H$ such that

$G \setminus H = o(L \setminus R)$ and $H \setminus G = o(R \setminus L)$.

Here, $G \setminus H$ and $H \setminus G$ represent the elements in $G$ and $H$ that are deleted and created by the transformation, respectively. Thus the first equation means that the transformation deletes exactly what is meant to be deleted by the rule, while the second equation states that it creates what should be created.

In order to check whether the provider fulfills the demands of the requester, their service models have to be matched.

In the next section we will discuss the notion of matching between service models.

4. PARTIAL MATCHING

The motivation behind the implementation of a service is to satisfy some requester requirements [4, 5]. When a requester finds and binds to a service, it has to be sure that the provider delivers its functionality in a way that is consistent with the assumptions about the required services made in the requester’s specifications [4, 5]. The requester’s assumptions are expressed as a goal, whereas provider’s guarantees for the implementation of services are expressed as offers.

Traditionally, an offer will satisfy a goal if the pre-conditions of the goal imply the pre-conditions of offer (so that, if a client invokes a service, this is actually executable) and the effects of the offer subsume that of the goal (so that the service performs the actions requested by the client) [5].

If a single goal is going to be satisfied by multiple offers then it will not be realistic to find the expected exact match. Instead, every offer will only provide a subset of the actions requested. On the other hand, by and large it could be a problem for a requester to know all of the pre-conditions to guarantee the excitability of the service.

For example, in Fig. 3 the provider assumes information on $fd$FlightDetail which is not guaranteed by requester, whereas requester asks for a reservation of a $h$Hotel which is not guaranteed by the offer. In an interactive booking process the requester may be perfectly happy to provide additional details (like preferred seating or other category of flight), so the assumption that the goal should contain a complete specification of the preconditions required by the provider seems unrealistic, even if it would guarantee absence of run-time errors. Hence, partial matching affects the relations between both preconditions and effects.

We say that a rule $p : L_p \rightarrow R_p$ is subsumed by a rule $q : L_q \rightarrow R_q$ if there is an occurrence $o : L_p \cup R_p \rightarrow L_q \cup R_q$ such that
\[ o(L_p \setminus R_p) \subseteq L_q \setminus R_q \text{ and } o(R_p \setminus L_p) = R_q \setminus L_q. \]

Again, \( L_q \setminus R_q \) and \( R_q \setminus L_q \) represent the elements deleted and created by rule \( q \), so that the first subset expresses that \( q \) deletes everything \( p \) does, but possibly more, and similarly for the second subset and creation of elements. There is a partial match between provider rule \( p \) (offer) and requestor rule \( r \) (goal), if there exists a common rule \( c : L_c \rightarrow R_c \) (representing all that is shared between \( p \) and \( r \)) such that \( r \) subsumes \( c \) and \( p \) subsumes \( c \).

5. INCREMENTAL PROCEDURE

When there is no single individual offer satisfactory for any goal, we move as:

1. Select an offer with a helpful partial match as explained in Section 4.
2. Create the remainder goal, which includes all requirements of the original goal not satisfied by the offer.
3. Iterate Step 1 and Step 2 until either the goal is fully satisfied or we have no further helpful offers.
4. Combine the individual offers into a single combined offer to receive feedback from the requester.

5.1 Construction of Remainder

After doing the helpful partial match, we create the remainder by decomposing the original goal into two rules, the sat rule and rem rule. Here, the sat rule contains the requirements already satisfied by the last offer while and the rem rule shows the remaining requirements.

We create the rule sat by applying the common rule \( c : L_c \rightarrow R_c \) to the left-hand side graph of the goal rule (as explained in Section 3). Recall that rule \( c \) contains the common parts (nodes and edges) of goal and the offer. As shown in Fig. 4 objects and edges in the offer rules with unshaded background represent common parts. Also, it is assumed that rule \( c \) contributes towards the goal (as mentioned in Step 1 above) because the matching is helpful (see Section 4). In our example, rule \( c \) creates \( f:Flight \), and \( :Receipt \) along with three edges.

More generally, if we have a goal rule \( r : L_r \rightarrow R_r \) then \( L_r \otimes_{c} I \rightarrow R_r \), with \( I \) derived from \( L_r \) by rule \( c \).

Now sat \( : L_{sat} \rightarrow R_{sat} \) is given by \( L_{sat} = L_r \) and \( R_{sat} = I \) while rem \( : L_{rem} \rightarrow R_{rem} \) is defined as \( L_{rem} = I = R_{sat} \), that is, the post-conditions of rule sat are the pre-conditions of rule rem and \( R_{rem} = R_r \), i.e., the post-conditions of the goal will remain the post-conditions of rem.
5.2 Individual and Combined Offers

After decomposing requirements in the course of matching, we move on to generate a combined offer for requester from the individual offers. The requester is then able to review this offer and decide whether they want to accept it, including possible additional or missing effects or preconditions.

The current scenario has two individual offers, i.e., Prov1 :: FlightRes and Prov2 :: HotelRes as shown in top part of Fig. 4. The combined offer can be generated by a construction known as that of the concurrent rule. Its application has the same overall effect like all the individual rules combined [22, 23]. The combined offer will thus integrate the effects of all the individual offers which participated in the partial matching.

Informally it can be written as

\[
L_{prov1} \cup (L_{prov2} \setminus L_{prov1}) \rightarrow R_{prov2} \cup (R_{prov1} \setminus L_{prov2})
\]

This is assuming consistency in identification of nodes and edges between the active offers as well as with goals—a condition which is not usually satisfied in a scenario where these are produced independently by different developers.

The complete formal definition of the concurrent rules accounts for this problem by allowing for explicit renaming of elements between rules, but follows the same intuitive idea: The preconditions of the combined offer can be generated by taking the union of preconditions of the first offer \(L_{prov1}\) and the those preconditions of the second offer which have not been produced by first offer \(L_{prov2} \setminus L_{prov1}\). Dually, the postcondition of the combined offer are given by the union of effects of the second offer with those effects of the first offer which are not used in preconditions of the second offer \(R_{prov1} \setminus L_{prov2}\).

For example in our case in Fig. 5, the pre-condition of the combined offer has only the nodes of \(L_{prov1}\) \((c:Client, cc:Credit Card, t:Trip)\) because \(L_{prov2}\) has already been achieved in \(R_{prov1}\). The postcondition of the combined offer will have \(R_{prov2}\) i.e. \((c:Client, cc:Credit Card, tp:Transport, h:Hotel, :Receipt, t:Trip)\) and only those nodes and edges of \(R_{prov1}\) which have not been used in \(L_{prov1}\) i.e. \((f:FlightDetail, f:Flight, :Receipt)\) as shown in Fig. 5.

6. RESULTS AND DISCUSSION

In this paper, we have proposed an approach for semi-automated composition of services using visual specifications of pre- and post conditions. The procedure is based on the repeated partial matching of provider offers with a requestor goal, which is reduced in the process until all requirements are satisfied or there are no more offers to consider. As a result, the procedure constructs a combined offer, which can be presented to the requestor to confirm if it is acceptable. The formalization of these notions and constructions is backed up by the theory of graph transformation. It is sketched here in a simplified form in order to give a precise yet readable account. In summary, the main theoretical results are as follows.

1. A definition of partial matching allowing to compare individual offers of services with the global goal of the requestor.
2. An incremental matching procedure based on the construction of a remainder of a goal with respect to a chosen partial match. Assuming a finite number of offers, the incremental matching procedure terminates. Thus, partial matching is decidable.
3. The combined offer, constructed as result of the matching, has the same overall effect of all the individual offer. That means, for each sequence of applications of individual offer rules there exists an application of the combined offer rule with the same effect, and vice versa.
7. REFERENCES


