Identifying Optimal Composite Services by Decomposing the Service Composition Problem

Accepted version of paper to be published in ICWS 2011. Copyright ©2011 IEEE.

Zachary J. Oster, Ganesh Ram Santhanam, and Samik Basu
Department of Computer Science, Iowa State University, Ames, Iowa 50011, USA
Email: {zjoster, gsanthan, sbasu}@iastate.edu

Abstract—For a Web service composition to satisfy a user’s needs, it must not only provide the desired functionality, but also have nonfunctional properties (e.g., reliability, availability, cost) that are acceptable to the user. In the recent past, several techniques have been developed and deployed to identify a composite service that conforms to the functional requirements and is also optimal with respect to the user-defined preferences over non-functional properties. However, these composition techniques are limited to using one formalism for specifying the required functionality; in short, the existing techniques cannot identify optimal (w.r.t. non-functional properties) composite services that are required to satisfy functional requirements described in multiple formalisms. We have previously proposed a meta-framework for service composition that involves decomposing the required functionality into a boolean combination of atomic requirements, which are expressed using different formalisms. This meta-framework supports the use of multiple formalisms and their corresponding composition algorithms within a single scenario. In this paper, we integrate support for unconditional preferences over nonfunctional requirements into this composition meta-framework. We show that for a large class of problems, local selection of preferred service(s) can yield the most preferred composite service that satisfies the desired functional requirements.

I. INTRODUCTION

An important feature of service-oriented architectures [1] is that they support the rapid development of new applications in the form of composite services (also known as compositions) using existing Web services. The process of assembling a composite service that satisfies the functional requirements of the user is called Web service composition [2]–[4].

In addition to satisfying the functional requirements of the user, a composite service should also respect the user’s preferences over non-functional properties of the desired composition, such as availability, cost, and throughput. For example, a travel booking service may be required to reserve airline tickets, rental cars, hotel rooms, or any combination of these (functional requirements). However, among many composite services that may achieve this requirement, the user may prefer one that has sufficient throughput to handle the expected message traffic and at the same time has low cost per use (non-functional properties). Similarly, among composite services that provide patients’ medical records, the user may prefer those that have a low response time to support fast decisions by physicians and strong security to satisfy legal requirements. In short, the overarching objective of service composition is to find the most preferred compositions among those that satisfy the functional requirements.

Many techniques for Web service composition use formal methods to model available services, functional requirements, non-functional properties, or some combination of these (see [2], [3] for a survey). Such techniques can formally guarantee that the resulting composite services provide the required functionality as well as an acceptable set of non-functional properties. However, most of these techniques require the user to specify the functional requirements in one particular type of semantics, which may not be sufficiently expressive to accurately represent all requirements for a given task. For example, the Roman model of Berardi et al. [6] uses labeled transition systems, while Lin et al. [7] use description logic [8] to model the functionality.

The “meta-framework” for service composition that we presented in [9] overcomes this problem by using an AND-OR tree to decompose the overall functional requirements into individual properties that may each be specified using different (possibly incompatible) semantics. The meta-framework realizes a composition by first identifying services that satisfy the individual properties using formalism-specific service selection methods and then verifying if a combination of these services satisfies the original requirements. However, this meta-framework does not allow for consideration of the user’s preferences over non-functional properties during the process of selecting components for the composition. Against this background, the primary objective of this paper is to develop formal methods that can be used in conjunction with our meta-framework to produce a set of compositions that (a) satisfy the functional requirements and (b) are most preferred with respect to the user’s preferences over non-functional attributes.

In recent years, many researchers have pursued methods for identifying compositions whose non-functional properties are optimal with respect to a set of user preferences, including [10]–[12]. However, this problem is known to

This work is supported in part by U.S. National Science Foundation grants CNS0709217 and CCF0702758.
be NP-hard for non-functional properties with quantitative valuations [12]. Additionally, composition methods that consider non-functional properties generally lack the flexibility that our meta-framework provides, i.e., using different formalisms to specify requirements and appropriate composition techniques. Therefore, there is a need for a technique that supports consideration of non-functional attributes while preserving the ability to use diverse composition methods and functional requirement specifications.

The contributions of this paper can be summarized as follows.

1) We propose a method for identifying preferred Web service compositions in the meta-framework proposed in [9], allowing functional requirements to be specified in multiple (possibly incompatible) formalisms.

2) Given a set of “atomic” properties that make up the overall functional requirement, our method performs analysis of preferences over the set of services satisfying the atomic properties in the functional requirement, as opposed to traditional methods that perform analysis of preferences over the set of services satisfying the overall functional requirement. That is, our method employs local selection of preferred services, while existing methods rely on global selection.

a) We prove that our local selection method will always identify a composition which satisfies functional requirements and is at least as preferred as any other composition that satisfies the functional requirements.

b) We show that our local selection method is at least as efficient as global selection methods; in fact, our method’s advantage increases as the problem becomes more complex and as additional services can be removed from consideration at intermediate steps.

3) Our method is not tightly coupled with any specific preference formalism and, therefore, can be used with any quantitative or qualitative preference formalism.

Organization: Section II introduces an example to be used for explaining concepts throughout this paper. Section III summarizes the existing work upon which our composition method is built. Section IV presents the details of our composition method. Section V gives proofs of correctness and a discussion of our method’s complexity. Section VI discusses related work. Section VII summarizes the paper and discusses future research directions.

II. ILLUSTRATIVE EXAMPLE

We will use the scenario for Web service-based delivery of rich multimedia content originally presented by Wagner and Kellerer in [13] (and also used in [12] and [11]) as a running example throughout this paper. In this scenario, the goal is to develop a Web service composition to accomplish the task of delivering multimedia content to a user’s electronic device, e.g., a mobile phone or Internet-connected television. This task involves four major subtasks, which are illustrated in Figure 1: transcoding, translation, merging, and compression.

For each subtask in this process, there may exist many Web services that can accomplish the subtask. Perhaps one compression service works well for sending content over the public Internet, but a different compression service performs better over a private wireless network. In addition, some text content providers may provide pre-translated text, eliminating the need for a separate translator; likewise, some video content providers might provide content that is already transcoded for the appropriate device. These alternative functional properties \( \varphi_1 \) are listed in Table I. To provide the overall functionality specified in Figure 1, it is necessary to choose a set of \( \varphi_i \) from Table I. The possible choices of \( \varphi_i \) that fulfill this requirement can be expressed symbolically as a Boolean combination of the \( \varphi_i \). For instance, \( \varphi_1 \land \varphi_2 \land \varphi_4 \land \varphi_5 \land \varphi_7 \land \varphi_8 \) or \( \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_7 \land \varphi_8 \) satisfy the required functionality in Figure 1.

Besides the functional requirements of the task, non-functional attributes play a major role in determining the preferred Web service composition for this application. The objective, in the presence of non-functional properties, is to identify a composite service that not only satisfies the given functional requirements but also is most preferred with respect to the non-functional properties of the services participating in the composition. This is an optimization problem. We focus on the three non-functional properties listed in Table II: cost, throughput (i.e., the rate at which the service can process and fulfill requests), and reliability. Table II also shows the domain of each property, i.e., the set of values that can be assigned to each service for that property. Preferences are described over these domains. For instance, lower cost is preferred to higher cost, while for throughput or reliability, high values are preferred to medium values, which are in turn preferred to low values. This is referred to as the intra-attribute preference.

It is vital for a useful Web service composition to both satisfy the given functional requirements and provide a
Table I
FUNCTIONAL REQUIREMENTS FOR MULTIMEDIA CONTENT DELIVERY SCENARIO

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>SUBTASK NAME</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>ψ₁</td>
<td>video</td>
<td>Retrieve requested video content (if any) from provider.</td>
</tr>
<tr>
<td>ψ₂</td>
<td>transcode</td>
<td>Transcode content for the appropriate device and network.</td>
</tr>
<tr>
<td>ψ₃</td>
<td>video and transcode</td>
<td>Retrieve a pre-transcoded copy of requested video from provider.</td>
</tr>
<tr>
<td>ψ₄</td>
<td>text</td>
<td>Retrieve requested text content (if any) from provider.</td>
</tr>
<tr>
<td>ψ₅</td>
<td>translate</td>
<td>Translate text content into the requested language.</td>
</tr>
<tr>
<td>ψ₆</td>
<td>text and translate</td>
<td>Retrieve a pre-translated copy of requested text from provider.</td>
</tr>
<tr>
<td>ψ₇</td>
<td>merge</td>
<td>Merge video and text content into one stream for transmission.</td>
</tr>
<tr>
<td>ψ₈</td>
<td>compress</td>
<td>Compress stream in appropriate format to send to user’s device.</td>
</tr>
</tbody>
</table>

Table II
NON-FUNCTIONAL ATTRIBUTES FOR MULTIMEDIA CONTENT DELIVERY SCENARIO

<table>
<thead>
<tr>
<th>ATTRIBUTE (ABBREV.)</th>
<th>TYPE</th>
<th>DOMAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (C)</td>
<td>Quantitative</td>
<td>Positive real numbers</td>
</tr>
<tr>
<td>Throughput (T)</td>
<td>Qualitative</td>
<td>[Low, Medium, High]</td>
</tr>
<tr>
<td>Reliability (R)</td>
<td>Qualitative</td>
<td>[Low, Medium, High]</td>
</tr>
</tbody>
</table>

preferred set of non-functional properties. In the following sections, we will show how our composition method provides such solutions.

III. BACKGROUND

As mentioned previously, there are two main challenges in solving the Web service composition problem. The first challenge, which was addressed in [9], is to deal with functional requirements that may be expressed using different and possibly incompatible formalisms. The second challenge is to analyze preferences over non-functional requirements to obtain a composition that is at least as preferred as any other possible composition with respect to those non-functional requirements. The focus of this paper is on solving the second challenge while taking advantage of the solution to the first challenge that was presented in [9]. In this section, we describe the meta-framework presented in [9] for dealing with functional requirements.

The meta-framework proposed in [9] is built upon the central assumption that the functional requirement θ can be decomposed into a set of individual atomic properties such that the boolean combination of the atomic properties is equivalent to θ. We specify such an atomic property by a tuple ψ ≜ ⟨φ, F⟩, where φ is an expression that is interpreted according to a formalism F. An example of an atomic property tuple is

⟨ϕ₈, F₈⟩ = ⟨A(!send U compress), CTL⟩

where A(!send U compress) (always fail to perform send until compress is performed) is a property expressed in the temporal logic formalism CTL (Computation Tree Logic [14]). The functional requirement θ is simply a boolean combination of atomic properties; we will also refer to θ as the composite property. For instance, in Table I, observe that satisfying requirement ψ₆ will produce the same result as satisfying the combination of requirements ψ₄ and ψ₅. To reflect this fact, we can write a composite property θ₄,₅,₆ ≜ (ψ₄ ∧ ψ₅) ∨ ψ₆.

We represent the decomposition of a composite property in our meta-framework using an AND-OR tree, where each leaf node is labeled with an atomic property and all non-leaf nodes are labeled with either AND (∧) or OR (∨). An AND node represents the conjunction of the functional properties represented by its child nodes, while an OR node indicates the disjunction of its child nodes’ functional properties. The root node of an AND-OR tree corresponds to the entire Boolean combination of functional requirements that must be satisfied by the requested composition. We write T° to indicate the AND-OR tree describing the Boolean decomposition of the property θ.

Example 1: Let θ represent the overall functional requirement for the multimedia provisioning scenario discussed in Section II. Figure 2 shows one possible realization of T°, changing the order in which atomic properties are considered can produce different realizations of T°.

The composition method presented in [9] uses T° both to enumerate all sets of atomic properties that are sufficient to fulfill the overall functional requirement θ and to identify all possible service compositions that are likely to satisfy one of these sufficient sets of atomic properties. In [9], once the sets of services that satisfy the atomic properties at the leaf-nodes of T° are identified, we convert the problem of identifying a service composition into that of satisfiability. Each atomic
service is viewed as a proposition, a composite service is viewed as a conjunction of propositions (each corresponding to an atomic service), and a set of services is viewed as a disjunctive normal form formula. The leaf nodes of $\mathcal{T}^\theta$ are labeled with disjunctive formulas, each OR-node is labeled with disjunctions of the labels of its child-nodes, and each AND-node is labeled with conjunctions of the labels of its child-nodes. A satisfiable assignment to the formula labeling the root of $\mathcal{T}^\theta$ corresponds to a set of propositions that hold true, i.e., a composite service containing the atomic services corresponding to these propositions. The set of satisfiable assignments, therefore, forms the set of candidate services corresponding to these propositions. The set of compositions that may conform to the requirements, i.e., a composite service containing the atomic services corresponding to these propositions. The set of compositions that are also preferred with respect to a set of preferences over the non-functional attributes of the compositions.

### IV. FINDING PREFERRED COMPOSITIONS

Among functionally feasible compositions, i.e., those that satisfy the functional requirements given in the form of an AND-OR tree, we are interested in finding those compositions that are also preferred with respect to a set of preferences over the non-functional attributes of the compositions.

#### A. Preferences over Non-Functional Attributes

We allow the user to specify his/her preferences over a set $X = \{X_1 \ldots X_m\}$ of non-functional attributes in the form of intra-attribute preferences with respect to each attribute $X_i$, i.e., a preference relation $\succ_i$ specifying an ordering over the possible values $D_i$ of each attribute. If $w_j(X_i)$ and $w_k(X_i)$ represent the valuations of $X_i$ in services $w_j$ and $w_k$ respectively, then we define $w_j(X_i) \succeq_i w_k(X_i)$ iff $w_j(X_i) \succ_i w_k(X_i) \lor w_j(X_i) = w_k(X_i)$.

For any service $w_i$, we denote its non-functional valuation as $V(w_i) = (w_i(X_1), \ldots, w_i(X_m))$ where $w_i(X_j)$ represents the value of the non-functional attribute $X_j$ in service $w_i$. We say that a service $w_i$ dominates (or is preferred to) another service $w_j$ if $V(w_i) \succ V(w_j)$, where $\succ$ is the dominance preference relation for comparing services with respect to the set of intra-attribute preference relations $\{\succ_1, \ldots, \succ_m\}$. To compute the dominance relation $\succ$, we use the semantics of Pareto dominance [15], in which $w_i$ is preferred to $w_j$ (denoted by $w_i \succ_p w_j$) if both of the following hold:

1. $\forall X_k \in X : w_i(X_k) \succeq_k w_j(X_k)$, i.e., $w_i$’s valuations for every non-functional attribute are preferred to or equal to the corresponding valuations of $w_j$.
2. There is at least one non-functional attribute $X_k$ for which $w_i(X_k) >_k w_j(X_k)$, i.e., $w_i$ is preferred (and not equal) to $w_j$.

Note that all non-functional properties are considered equally important in this semantics.

Given a set of preferences over the non-functional attributes of a set $C = \{C_1 \ldots C_n\}$ of composition, we call the set $W^n_{C'} = \{C_i \mid \nexists C_j : C_j \models \varphi \land C_j \succ C_i\}$ the non-dominated or most preferred set of compositions in $C$, i.e., there is no feasible composition that is preferred to any of the most preferred compositions.

**Example 2:** Table III displays sets of services that satisfy each atomic property at the leaf level of $\mathcal{T}^\theta$, as presented in Figure 2, along with each service’s valuations for three non-functional properties: cost, throughput, and reliability. (The possible valuations for each property are given in Table II.) Consider the services $w_51$ and $w_52$ shown in Table III. We observe that the cost of $w_51$ is preferred to $w_52$, the throughput of $w_52$ is preferred to $w_51$, and both $w_51$ and $w_52$ have equal reliability. Since neither service satisfies the criteria for Pareto dominance, the non-dominated set will be $\{w_51, w_52\}$. However, if throughput is not considered when determining dominance, then $w_51$ dominates $w_52$.

In place of Pareto dominance, one could use other, more expressive preference languages for specifying the user preferences and the corresponding semantics for the dominance preference [16] to select preferred services along with our approach. However, for simplicity, we will consider only Pareto dominance in this paper. We also note that the above chosen preference specification language and dominance semantics can be used to represent and reason about both qualitative and quantitative preferences.

#### B. Computing Non-dominated Compositions

We now present Algorithm 1, which finds the non-dominated set of feasible compositions with respect to (a) a set of functional requirements $\theta$ expressed using our meta-framework from [9] and (b) a set of preferences over the non-functional attributes. The basic idea of this algorithm is to first identify a set of locally feasible services that satisfy the atomic requirement at each leaf node in the AND-OR tree and then build preferred compositions of such locally feasible services by traversing the AND-OR tree.
Algorithm 1 Optimized composition method for the case where services are compatible

1: procedure COMPACTCOMP(qk)
2: if qk is a leaf node then $\triangleright$ labeled with $\langle \varphi_i, F_i \rangle$
3: $W_k := M_i (\langle \varphi_i, F_i \rangle)$
4: for all $w_{ki} \in W_k$ do
5: if $\nexists w_{kj} \in W_k : w_{kj} \succ w_{ki}$ then
6: $W_{ki}^{nd} := W_{ki}^{nd} \cup \{w_{ki}\}$
7: end if
8: end for
9: else if $q_k$ is an AND node then
10: $W_{k_1} := COMPACTCOMP(children(q_k)[1])$
11: $W_{k_2} := COMPACTCOMP(children(q_k)[2])$
12: $W_k := \{w_{k_1} \otimes w_{k_2} : w_{k_1} \in W_{k_1} \land w_{k_2} \in W_{k_2}\}$
13: for all $w_{ki} \in W_k$ do
14: if $\nexists w_{kj} \in W_k : w_{kj} \succ w_{ki}$ then
15: $W_{ki}^{nd} := W_{ki}^{nd} \cup \{w_{ki}\}$
16: end if
17: end for
18: else if $q_k$ is an OR node then
19: $W_{k_1} := COMPACTCOMP(children(q_k)[1])$
20: $W_{k_2} := COMPACTCOMP(children(q_k)[2])$
21: for all $w_{ki \_1} \in W_{k_1}$ do
22: if $\nexists w_{kj} \in W_{k_2} : w_{kj} \succ w_{ki \_1}$ then
23: $W_{ki}^{nd} := W_{ki}^{nd} \cup \{w_{ki \_1}\}$
24: end if
25: end for
26: for all $w_{ki \_2} \in W_{k_2}$ do
27: if $\nexists w_{kj} \in W_{k_1} : w_{kj} \succ w_{ki \_2}$ then
28: $W_{ki}^{nd} := W_{ki}^{nd} \cup \{w_{ki \_2}\}$
29: end if
30: end for
31: end if
32: return $W_k^{nd}$
33: end procedure

Algorithm 1 assumes that all services in the repository $R$ are compatible, i.e., any service in $R$ can be successfully composed with any other service in $R$ without compromising the functionality of either component service. Hence, when computing the possible compositions at each node, Algorithm 1 only considers the non-dominated sets of feasible services of each child node (i.e., a local selection approach). This algorithm is useful in intra-organizational settings where all services are typically interoperable. Note that if this compatibility assumption does not hold, it is still possible to find a set of preferred compositions using our meta-framework, although it is then necessary to consider all possible compositions of the services identified at each child node instead of using a local selection approach.

Algorithm 1 traverses the AND-OR tree $T^0$ from the bottom up, identifying services that satisfy the atomic functional property at each leaf node and then forming compositions of services that satisfy the overall functional property represented by each AND or OR node. After the traversal is completed, Algorithm 1 returns a set $W_0^{nd}$ of non-dominated compositions (corresponding to the root node $q_0$). Note that if any component service $w_{ij}$ in a composition $C$ in $W_0^{nd}$ satisfies an atomic property $\langle \varphi_i, F_i \rangle$, then the entire composition $C$ also satisfies $\langle \varphi_i, F_i \rangle$ by our assumption that all services in the repository $R$ are compatible.

Let $\varphi_i$ be an atomic property represented by a leaf node in the tree. Algorithm 1 first identifies the set $W_i$ of services that satisfy $\langle \varphi_i, F_i \rangle$ (Line 3). For each atomic property $\langle \varphi_i, F_i \rangle$ represented by a leaf node in the tree, we assume the existence of a procedure $M_i$ that can identify a set $W_i$ of services that satisfy $\langle \varphi_i, F_i \rangle$ and can later be used to verify that a preferred composition satisfies $\langle \varphi_i, F_i \rangle$. We abstract the details of $M_i$, so that different service discovery mechanisms can be used corresponding to the requirements expressed in different formalisms. Subsequently, the algorithm computes the non-dominated set $W_i^{nd}$ of services (Lines 4-8) with respect to the user’s preferences over the non-functional attributes. This non-dominated service set is then passed on to the parent of the leaf node (Line 32).

Example 3: Each service shown in Table III can be identified by using an existing compositional method $M_i$ that can address the composition problem if the functional requirements are provided in one specific formalism $F_i$. Consider the leaf node $q_5$, which corresponds to the atomic property $\langle \varphi_5, F_5 \rangle$. The existing method $M_5$ returns the set of feasible services $W_5 = \{w_{51}, w_{52}\}$. Recall from Example 2 that applying the witness-based dominance relation from [17] over the set $W_5$ produces the non-dominated service set $W_5^{nd} = \{w_{51}, w_{52}\}$. Since neither service dominates the other, both services are passed to the parent node of $q_5$.

When Algorithm 1 encounters an AND node, it identifies all compositions that can be created by selecting one service each from the non-dominated solution sets $W_{k_1}$ and $W_{k_2}$ received from the two child nodes (Lines 10-12). We use a generic composition operator $\otimes$ in order to abstract the details of whether the composition of services is in parallel or serial. We compute the non-functional attribute valuation of a composition in terms of the non-functional attributes of its constituents. In this paper, we use the worst-frontier [10] method for aggregating the valuations of the component services, where the valuation of an attribute $X_k$ in a composition of two services $w_i$ and $w_j$ is the worst among their respective valuations, i.e., $\min\{w_i(X_k), w_j(X_k)\}$. Hence, the less-preferred valuation(s) from either of the child nodes’ services becomes the valuation for the composition. However, other methods of computing the valuation of a composition can be used, such as maximum (e.g., in the case of throughputs), sum (e.g., for cost), etc. as appropriate. After all possible compositions $W_k$ have been identified and their resulting non-functional property valuations have been...
Table IV

<table>
<thead>
<tr>
<th>ID</th>
<th>STRUCTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_{c1}</td>
<td>w_{11} \otimes w_{22} \otimes w_{33} \otimes w_{51} \otimes w_{71} \otimes w_{92}</td>
</tr>
<tr>
<td>w_{c2}</td>
<td>w_{12} \otimes w_{22} \otimes w_{32} \otimes w_{71} \otimes w_{81}</td>
</tr>
<tr>
<td>w_{c3}</td>
<td>w_{12} \otimes w_{23} \otimes w_{33} \otimes w_{51} \otimes w_{71} \otimes w_{92}</td>
</tr>
<tr>
<td>w_{c4}</td>
<td>w_{31} \otimes w_{61} \otimes w_{71} \otimes w_{81}</td>
</tr>
<tr>
<td>w_{c5}</td>
<td>w_{32} \otimes w_{41} \otimes w_{51} \otimes w_{71} \otimes w_{92}</td>
</tr>
</tbody>
</table>

computed by using an appropriate aggregation method, Algorithm 1 computes the non-dominated set \( W_k^{nd} \) of solutions (Lines 13-17) and passes it on to the current AND node’s parent unless the current node is the root node (Line 32).

At each OR node, Algorithm 1 separately considers the sets of non-dominated services \( W_{k1} \) and \( W_{k2} \) that are feasible and non-dominated at the child nodes. The algorithm first tests each service \( w_{k1} \) in \( W_{k1} \) to determine whether it is dominated by any service in \( W_{k2} \). If \( w_{k1} \) is not dominated by any service in \( W_{k2} \), then \( w_{k1} \) is added to the OR node’s set of non-dominated services \( W_k^{nd} \) (Lines 21-25). Similarly, each service \( w_{k2} \) in \( W_{k2} \) is tested against every service in \( W_{k1} \); if no service in \( W_{k1} \) dominates \( w_{k2} \), then \( w_{k2} \) is added to the non-dominated service set \( W_k^{nd} \) for this OR node (Lines 26-30). This ensures that only the services that are non-dominated in \( W_{k1} \cup W_{k2} \) are added to \( W_k^{nd} \). Since services are considered individually and no new compositions are considered, there is no need to aggregate the non-functional property valuations of different services. Algorithm 1 then returns the set \( W_k^{nd} \) of solutions to the OR node’s parent unless the current node is the root node (Line 32).

Example 4: Table IV lists the set of non-dominated services that are returned by Algorithm 1 after traversing the AND-OR tree \( T^d \). These services are generated from the locally optimal solutions identified using the data from Table III and the Pareto dominance relation discussed in the previous subsection. Table IV also shows the non-functional property valuations for each listed composition, which are computed by summing the components’ costs and applying the worst-frontier heuristic over the throughput and reliability valuations of the components. All of these compositions fully satisfy the overall functional requirement \( \theta \) (as specified by our compatibility assumption), but none of them can be considered globally optimal according to the Pareto dominance relation.

V. THEORETICAL PROPERTIES

We begin this section by reiterating the compatibility assumption stated previously for Algorithm 1, i.e., that any service in the repository \( R \) may be freely composed with any other service in \( R \). Given this assumption, we prove that Algorithm 1 is sound and weakly complete. Let \( g_0 \) be the root node of the AND-OR tree \( T^\theta \); then \( W_0^{nd} \) is the set of non-dominated services returned by Algorithm 1 for a given composition problem.

Theorem 1 (Soundness): If all services in \( R \) are compatible with each other, then \( \forall w_c \in W_0^{nd} : w_c \models \theta \land \left( \not\exists w'_c : w'_c \models \theta \land \lnot w'_c \succeq_p w_c \right) \).

Proof: The compatibility assumption ensures that any composition returned by Algorithm 1 satisfies \( \theta \). To prove the rest of the theorem, we proceed by induction on the height of the AND-OR tree \( T^\theta \). The theorem holds at each leaf node \( q_k \), since only the set \( W_k^{nd} \) of non-dominated services that satisfy the atomic property at that leaf node is passed on to the parent node; therefore, no non-dominated service is ever returned. At every level \( \ell \) above the leaf-level, every node of \( T^\theta \) is either an AND node or an OR node.

Each OR node \( q_k \) at level \( \ell \) receives a set of non-dominated services from both of its child nodes; we denote these sets by \( W_{k1} \) and \( W_{k2} \). Assume that the theorem holds at level \( \ell + 1 \); then no service \( w_{k1} \) in \( W_{k1} \) (resp. \( w_{k2} \) in \( W_{k2} \)) is dominated by any other service \( w_{k1j} \) in \( W_{k1j} \) (resp. \( w_{k2j} \) in \( W_{k2j} \)). Each service \( w_{k1j} \) in \( W_{k1} \) is added to the non-dominated service set \( W_k^{nd} \) only if it is not dominated by any service in \( W_{k2j} \); likewise, each service \( w_{k2j} \) in \( W_{k2} \) is added to \( W_k^{nd} \) only if it is not dominated by any service in \( W_{k1} \). As a result, no service that appears in \( W_k^{nd} \) is dominated by any other service in \( W_{k1} \cup W_{k2} \), so the theorem holds.

For the AND nodes, again assume that the theorem holds at level \( \ell + 1 \). Consider an arbitrary AND node \( q_k \) at level \( \ell \) and a composition \( w'_{a} = w'_{a1} \odot w'_{a2} \) where \( w'_{a1} \) and \( w'_{a2} \) are non-dominated at \( q_k \)’s child nodes \( q_{k1} \) and \( q_{k2} \), respectively (i.e., \( w'_{a} \in W_{k1} \) and \( w'_{a} \in W_{k2} \)). Suppose in contradiction that Algorithm 1 returns a different composition \( w_a \), which is preferred to \( w'_{a} \) by Pareto dominance (i.e., \( w_a >_p w'_{a} \)) but which contains at least one service that is not contained in either child node’s non-dominated set \( W_{k1} \) or \( W_{k2} \). Let \( w_a = w_a \odot w'_b \), where \( w'_b \notin W_{k1} \). Note that \( w_a \) is Pareto-dominated by \( w'_{a} \) i.e., \( w'_a \succ_p w_a \). By the definition of Pareto dominance, there is at least one non-functional property \( X_j \) for which \( w_a(X_j) \succ_j w'_a(X_j) \); this implies that \( (w'_a \odot w'_b)(X_j) \succ_j (w_a \odot w'_b)(X_j) \). Using worst-frontier aggregation, this is equivalent to

\[
\min\{w_a(X_j) \cup w'_b(X_j)\} \succ_j \min\{w'_a(X_j) \cup w'_b(X_j)\}
\]

Additionally, because \( w'_a \succ_p w_a \), it is clear that

\[
w'_a(X_j) \prec_j w_a(X_j)
\]

If \( w'_a(X_j) = w_a(X_j) \), then \( \min\{w_a(X_j) \cup w'_b(X_j)\} = \min\{w'_a(X_j) \cup w'_b(X_j)\} \), so Equation 1 is violated, which is a contradiction. Suppose instead that \( w'_a(X_j) \succ_j w_a(X_j) \).

By Equation 2, there are two cases to consider.

1) If \( w_a(X_j) \succ_j w'_b(X_j) \) or if \( w'_b(X_j) \succ_j w_a(X_j) \), then

\[
\min\{w'_a(X_j) \cup w'_b(X_j)\} = w'_a(X_j)
\]

and by Equation 2, \( w'_a(X_j) \succ_j w_a(X_j) \).

2) If \( w_a(X_j) \not\succ_j w'_b(X_j) \) and if \( w'_b(X_j) \not\succ_j w_a(X_j) \), then

\[
\min\{w'_a(X_j) \cup w'_b(X_j)\} = w'_b(X_j)
\]

and by Equation 2, \( w'_b(X_j) \succ_j w_a(X_j) \).

\[\square\]
In both cases, Equation 1 is violated, which is a contradiction. Therefore, the theorem holds at all AND nodes at level \( \ell \).

**Theorem 2 (Weak Completeness):** If all services in \( R \) are compatible with each other and if \( W^* = \{ w_e : w_e \models \theta \} \), then \( W^* \neq \emptyset \Rightarrow W_0^{nd} \neq \emptyset \wedge W_2^{nd} \subseteq W^* \).

**Proof:** The proof proceeds by induction on the height of the AND-OR tree \( T^\theta \). Suppose there is a feasible composition \( w_e \in W^* \). For each leaf node \( q_k \), by the definition of Pareto dominance, \( W_k \neq \emptyset \Rightarrow W_k^{nd} \neq \emptyset \) holds trivially. Every non-leaf node at level \( \ell \) of \( T^\theta \) is either an AND node or an OR node. Suppose the theorem holds for every node at level \( \ell + 1 \). Consider a node \( q_k \) at level \( \ell \) with children \( q_{k1} \) and \( q_{k2} \) that return sets of services \( W_{k1} \) and \( W_{k2} \) respectively. If \( q_k \) is an OR node, then \( W^* \neq \emptyset \) implies that at least one of \( W_{k1} \) or \( W_{k2} \) is non-empty (otherwise, the functional requirement at \( q_k \) cannot be satisfied). Hence \( W_{k1} \cup W_{k2} \neq \emptyset \Rightarrow W_{k2}^{nd} \neq \emptyset \) (by the definition of Pareto dominance). Likewise, if \( q_k \) is an AND node, then \( W^* \neq \emptyset \) implies that both \( W_{k1} \) and \( W_{k2} \) are non-empty. Because all services are compatible, \( W_{k1} \neq \emptyset \wedge W_{k2} \neq \emptyset \Rightarrow W_k \neq \emptyset \Rightarrow W_k^{nd} \neq \emptyset \) (by the definition of Pareto dominance). Hence at every node \( q_k \) in \( T^\theta \) (including the root node), \( W_k^{nd} \neq \emptyset \) if there is at least one service that satisfies the required functionality in \( q_k \). Therefore, Algorithm 1 always produces a composition that satisfies the overall functional requirement \( \theta \) if such a composition exists. In addition, by Theorem 1, this composition is non-dominated with respect to the user’s preferences over non-functional properties.

The complexity of Algorithm 1 is determined largely by the complexity of the methods \( \mathcal{M}_i \) used to identify services that satisfy each atomic property \( \langle \varphi_i, F_i \rangle \) and the dominance testing that is required to identify the most preferred compositions. Since the specific methods \( \mathcal{M}_i \) and the particular dominance relation may be different for each application of Algorithm 1, we focus on the required number of dominance tests. This depends on the number of atomic properties \( \langle \varphi_i, F_i \rangle \) representing \( \theta \) (denoted by \( n \)), the maximum number of services in \( R \) satisfying any one \( \langle \varphi_i, F_i \rangle \) (denoted by \( k \)), and the maximum fraction of services that are not removed from consideration by any node’s children (denoted by \( d \), where \( 0 < d \leq 1 \)).

At most \( \frac{2}{3}(k)(k-1) \) dominance tests are needed to compute the non-dominated service sets at the leaf nodes. The number of additional dominance tests depends on the proportion of AND nodes versus OR nodes in the AND-OR tree: a tree where all non-leaf nodes are OR nodes requires the least dominance tests, while a tree where all non-leaf nodes are AND nodes requires the most. For a tree with all OR nodes above the leaf level, an additional \( \frac{2}{3}(k)(k-1) \) dominance tests are required; if all non-leaf nodes are AND nodes, then an additional \( \sum_{i=1}^{n} \frac{n}{n^i} \binom{(dk)^2}{(dk)^2 - 1} \) dominance tests are needed. In contrast, a global selection approach that simply identifies and performs dominance testing over all feasible compositions would require between \( k(k-1) \) tests (if all non-leaf nodes are OR nodes) and \( \frac{2}{3}(k)(k-1) + \frac{1}{3}(k^n)(k^n-1) \) tests (if all non-leaf nodes are AND nodes), including the tests required at the leaf nodes. Our local selection-based method is therefore at least as efficient as global selection-based methods; furthermore, our method’s efficiency advantage improves as the number of atomic properties increases and as more services are removed from consideration at early stages of the algorithm.

VI. RELATED WORK

Two existing methods for Web service composition are similar to our method. The LOEM method proposed in [11] works by identifying services for individual subtasks, applying local selection to identify several services for each subtask that are likely to produce a preferred composition, enumerating all possible compositions that can be created from the services identified for each subtask, and then choosing the best composition using integer programming (MIP). Unlike our method, which decomposes the functional requirements for a composition and applies MIP to determine the optimal composition to solve the given problem. However, there are major differences between our method and these other methods. Both [12] and [11] deal specifically with quality-of-service (QoS) properties, while our method is capable of handling any type of non-functional property (including QoS properties). In addition, although [11] discretizes services’ non-functional property valuations into a finite number of ranges, the methods given in [12] and [11] are restricted to using only quantitative valuations for non-functional properties, since compositions are selected by maximizing a utility function.

Our method supports the use of both qualitative and quantitative non-functional property valuations, as it employs a more general dominance testing framework that has been proven to be capable of identifying the best alternative under these conditions. Along similar lines, TCP-Compose* [10] uses TCP-nets [16] to model conditional preferences between different qualitative non-functional properties and identify compositions that are non-dominated with respect to a set of user-specified preferences over non-functional properties. Like other composition methods, TCP-Compose* is not capable of handling functional requirements specified using multiple formalisms. However, it would be possible to use TCP-nets to specify the user’s non-functional attribute.
preferences within the context of the algorithms presented in this paper, as our algorithms do not depend on any one formalism for specifying preferences.

VII. DISCUSSION

Many methods for Web service composition model services and requirements using formal methods, but most of the existing methods that do so require the use of one specific formalism, limiting the scope of the composition problems that can be considered. We have previously proposed a meta-framework for service composition in [9] that overcomes this limitation; however, this meta-framework does not support consideration of non-functional properties. In this paper, we have enhanced the composition metawork proposed in [9] by selecting component services according to their non-functional properties so that no other composition that satisfies the given functional requirements has a set of non-functional properties that is preferred to the composition produced by our method.

An immediate objective for future work is to implement a simulation framework for our composition method in order to empirically compare its performance to similar composition algorithms, such as the ones presented in [11] and [12]. Another factor that we will consider is the effectiveness of different dominance relations for choosing preferred component services within our method. In addition, we plan to apply the principles of our composition approach to the problems of Web service substitution and adaptation, as we believe support for reuse of intermediate results may produce significant improvements in performance compared to current methods. We also plan to explore the use of efficient techniques for determining AND-OR graph satisfaction, such as those in [18] and [19]. Finally, we intend to consider whether different configurations of the AND-OR tree affect the preferred service composition identified by our method.

REFERENCES


