On Context-Specific Substitutability of Web Services

Jyotishman Pathak **Samik Basu** **Vasant Honavar**
Department of Computer Science
Iowa State University, Ames, IA 50011-1040, USA
{jpathak,sbasu,honavar}@cs.iastate.edu

Abstract

Web service substitution refers to the problem of identifying a service that can replace another service in the context of a composition with a specified functionality. Existing solutions to this problem rely on detecting the functional and behavioral equivalence of a particular service to be replaced and candidate services that could replace it. We introduce the notion of context-specific substitutability, where context refers to the overall functionality of the composition that is required to be maintained after replacement of its constituents. Using the context information, we investigate two variants of the substitution problem, namely environment-independent and environment-dependent, where environment refers to the constituents of a composition and show how the substitutability criteria can be relaxed within this model. We provide a logical formulation of the resulting criteria based on model checking techniques as well as prove the soundness and completeness of the proposed approach.

Keywords. Web services, Composition, Substitution, Labeled Transition Systems, Mu-Calculus, Model Checking

1 Introduction

Service-oriented computing offers a powerful approach for the construction of complex applications from autonomously developed, distributed software components in multiple domains including e-Science, e-Business and e-Government. Consequently, there is a growing body of work focused on various aspects of specification, verification and composition of Web services.[1]

However, assembling a composite service that satisfy a desired set of requirements is only the first step. Ensuring that a composite service, once assembled, can be successfully deployed presents additional challenges that need to be addressed. Suppose a composite service Q relies on component/pre-existing services $Q_1 \cdots Q_n$. Consider a scenario wherein one of the component services, say Q_1, becomes unavailable either because the service provider for Q_1 chooses not to offer it any more or updates it (e.g., by adding/removing some of Q_1's features), thereby altering its behavior. Consequently, the behavior of the composite service Q that relies on Q_1 is also altered. Because assembly of composite services in general is computationally costly, it is desirable to replace only the affected component(s) e.g. Q_1, with an alternative, say $Q_1^{'}$, while ensuring that the resulting composite service $Q^{'}$ obtained by replacing Q_1 with $Q_1^{'}$ can support (minimally) all of the functionality that was originally offered by Q.

As a result, identifying a component service that can substitute for another service has become an important problem in service-oriented computing. Of particular interest is the problem of determining whether a service can be replaced by another service in a specific *context* (or *property*) φ, which essentially refers to the functionality of the composition that must be preserved after the substitution. Previous solutions [4, 6, 11, 12] to this problem have relied on establishing functional or behavioral equivalence between the service that is being replaced and the replacement service (refer to section 6 for details).

We note that the requirement of functional/behavioral equivalence is stronger than that is often needed in practice for substituting one service with another. Hence, we introduce two variants of the *context-specific service substitutability* problem that are based on weaker and flexible requirements than those assumed by previous approaches. The solution makes it possible to safely replace a service Q_1 with $Q_1^{'}$ within the context of a given composition, even though $Q_1^{'}$ may not meet the stronger requirement of being functionally or behaviorally equivalent to Q_1. More precisely, we represent a composition (denoted by $||$) of two services Q_1 and Q_2 that realizes a specific functionality or property (denoted by φ and expressed in temporal logic) by $Q_1 \,||\, Q_2 \models \varphi$. In the event Q_1 becomes unavailable, the goal is to identify candidates (Q_1') that can be used as replacement for Q_1 in the *environment* Q_2 and *property* φ. We represent services in our setting as labeled transition systems [14] and properties by mu-calculus [8] formulas, and introduce the notion of *quotienting* such formulas. Informally, quotienting can be regarded as "factoring" an existing property φ by a system (Web services in our case), to yield another property ψ (in the same logic as φ). We show how the quotienting technique can be used to identify a substitute for another service within the specific environment and context of a particular composition.

[1] In this paper, we use the terms "service" and "Web service" interchangeably.

The main contributions of this paper are as follows:

1. We introduce two variants of the problem of context-specific substitution of Web services, namely environment-dependent and environment-independent, that relaxes the stronger requirements (e.g., simulation/bisimulation equivalences) for replacing services prevalent in existing approaches.
2. We present an algorithm to determine whether a service can be replaced by another in a much more flexible setting using a technique called quotienting. The quotienting operation is defined in a manner that allows us to seamlessly take into consideration possible nondeterminism in the service behavior.
3. We establish the soundness and completeness of the proposed solution to determine context-specific substitutability of Web services.

Organization. The rest of the paper is organized as follows: Section 2 gives an overview of our approach and provides an example to illustrate the main ideas. Section 3 briefly introduces the relevant concepts from labeled transition systems and mu-calculus needed in the rest of the paper. Section 4 describes the quotienting rules for mu-calculus. Section 5 shows how the problems of determining context-specific substitutability of a service can be reduced to satisfiability of the quotiented mu-calculus formulae. The paper concludes with a discussion of related work in Section 6 followed by a summary of the main results and a brief outline of some directions for further research in Section 7.

2 Overview of our approach

2.1 Problem Definition

The problem of determining whether a service can be substituted by another can have two different variants and can be stated as follows.

Environment-Independent Substitutability: *Given a property φ, and services Q_1 and Q_1', can Q_1' substitute Q_1 regardless of the environment of Q_1?* Formally, for specific Q_1, Q_1' and φ:

$$\forall Q_2 : (Q_1 \mid\mid Q_2 \models \varphi) \stackrel{?}{\Rightarrow} (Q_1' \mid\mid Q_2 \models \varphi) \qquad (1)$$

i.e. can Q_1' replace Q_1 such that, when Q_1' is composed with *any* Q_2,[2] the resulting composition realizes φ? Observe that we only consider the Q_2s where $Q_1 \mid\mid Q_2 \models \varphi$. Compositions with Q_2s for which $Q_1 \mid\mid Q_2 \not\models \varphi$ (i.e., compositions that do not satisfy a desired property) are not interesting; the antecedent of the implication is false leading to satisfiability of the formula in equation 1.

Note that this notion of substitutability is applicable in the setting where it is unknown apriori the services with whom the substitute (Q_1') is going to interact, and hence it is useful to guarantee that the substitute can minimally interact with *any* service (i.e., the environment) that can interact with the original (Q_1) [6]. A relaxed version of the above problem is where we consider the substitution only for a *particular* environment that is known apriori; a problem that we refer to as environment-dependent substitutability.

Environment-Dependent Substitutability: *Given a property φ, and services Q_1 and Q_1', can Q_1' substitute Q_1 for a specific environment of Q_1?* I.e., for a particular Q_1, Q_1', Q_2 and φ, does $Q_1 \mid\mid Q_2 \models \varphi$ imply $Q_1' \mid\mid Q_2 \models \varphi$? Notationally,

$$(Q_1 \mid\mid Q_2 \models \varphi) \stackrel{?}{\Rightarrow} (Q_1' \mid\mid Q_2 \models \varphi) \qquad (2)$$

Note that environment-independent substitutability implies environment-dependent substitutability, but not the other way around. Thus, the solution set for the latter is a superset of the solution set for the former.

2.2 Proposed Solution

To address the problems defined in equations 1 and 2, we will use the technique of *quotienting*. As outlined above, quotienting of a property φ by Q, denoted by (φ / Q), results in a property ψ (in the same logic as φ) which if satisfied by Q' leads to $Q \mid\mid Q' \models \varphi$. Formally:

$$\forall Q' : (Q \mid\mid Q' \models \varphi) \Leftrightarrow (Q' \models (\varphi / Q))$$

Quotienting operation, therefore, captures the (temporal) *obligation* imposed by Q on its environment (Q') in order to satisfy φ.

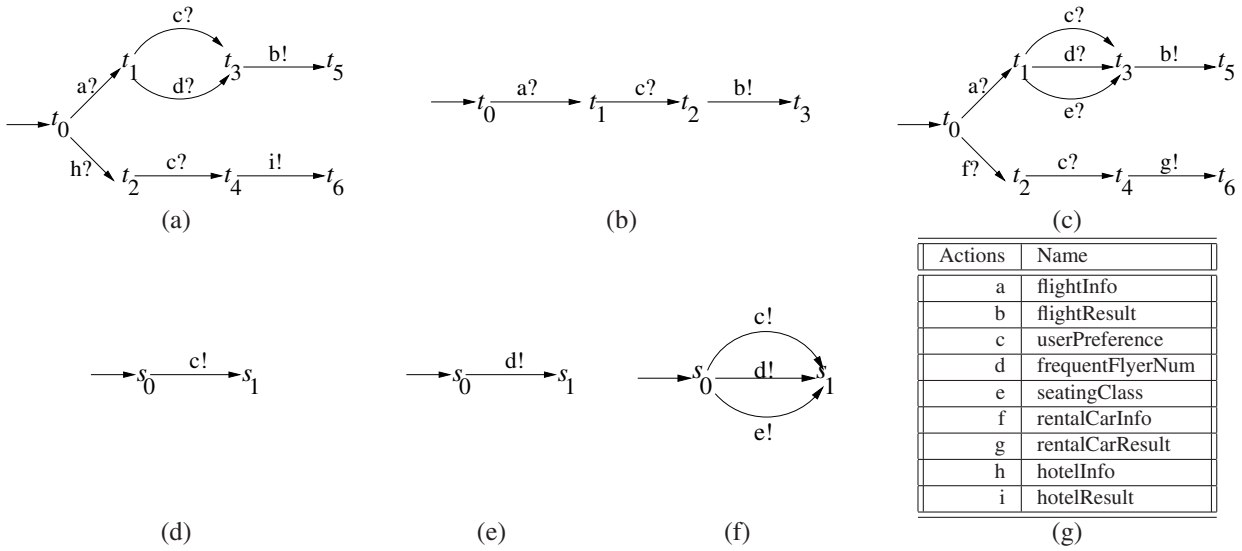
Going back to equation 1, the result of (φ / Q_1) denotes the property that must be satisfied by all the possible Q_2s such that $Q_1 \mid\mid Q_2 \models \varphi$. Similarly, (φ / Q_1') must be satisfied by all the services (say Q_2') such that $Q_1' \mid\mid Q_2' \models \varphi$. Proceeding further, if the set of Q_2s is a subset of set Q_2's, then in all environments of Q_1 where φ is satisfied, Q_1' when placed in those environments also satisfies φ. Therefore, the problem in equation 1 can be reduced to satisfiability (model checking [8]) of $(\varphi / Q_1) \Rightarrow (\varphi / Q_1')$.

Next consider the problem in equation 2. Here, (φ / Q_2) is the property that Q_1 satisfies when $Q_1 \mid\mid Q_2 \models \varphi$. In other words, Q_1' must also satisfy (φ / Q_2) in order to be able to substitute Q_1 in the context of Q_2 and φ. The substitutability problem, therefore, can be reduced to satisfiability of (φ / Q_2) by Q_1'. I.e. Solution to equation 2 holds if and only if $Q_1' \models (\varphi / Q_2)$.

2.3 Illustrative Example

Consider a setting wherein a traveler is interested in getting information about airline reservations by interacting with an existing Web service called `FunTravel`. This service is composed of two component services namely, `TravelSearch` (denoted by Q_1) and `ProfileInfo` (denoted by Q_2). Q_1 allows its clients to search for flight tickets as well as hotel rooms, whereas Q_2 stores and provides personal profile information (e.g., airline/hotel pref-

[2] In this case, Q_2 becomes the environment.



Actions	Name
a	flightInfo
b	flightResult
c	userPreference
d	frequentFlyerNum
e	seatingClass
f	rentalCarInfo
g	rentalCarResult
h	hotelInfo
i	hotelResult

(g)

Figure 1: Sample Services. (a) Q_1. (b) Q_1'. (c) Q_1''. (d) Q_2. (e) Q_2'. (f) Q_2''. (g) Action-Name mapping.

erences) of its clients. An interaction between the client and `FunTravel` (and the two component services) can be described as follows: (i) First the client sends a message to Q_1 to search for a flight with required inputs (e.g., email address, departure/arrival cities); (ii) On receipt of the message, Q_1 interacts with Q_2 to retrieve client's profile information (e.g., airline preference); (iii) Once this information is received, Q_1 searches for available flight options, and sends the search results back to the client. Thus, the functionality (or *property* denoted by φ) realized by this composition is: *given an input for searching flight reservations, the composite service provides a list of available options* (if any). We will show later (section 3.2) how to represent such properties in temporal logic using mu-calculus formulas.

The services in our model are represented using Labeled Transition Systems[3] (LTS, see section 3.1) where the states of the LTS correspond to various configurations of the service, whereas transitions between the states correspond to how the service evolves by updating its configuration. The transitions are labeled by input (denoted by "?") and output ("!") actions over which a service can synchronize, and corresponds to an event in which the service receives and sends a message, respectively. Figures 1(a) & 1(d) shows the LTS representation of Q_1 and Q_2, respectively, whereas Figure 2 show their composition. Note that, it is possible to compose Q_1 with Q_2' (Figure 1(e)) and Q_2'' (Figure 1(f)) as well because both $Q_1 \mid\mid Q_2' \models \varphi$ and $Q_1 \mid\mid Q_2'' \models \varphi$.

Now, assume that Q_1 becomes unavailable and needs to be substituted. As per equation 2, Q_1' (Figure 1(b)) which allows only to search for flight reservations, can act as a candidate replacement and can be composed with Q_2 (i.e., the environment) to satisfy the property φ. However, it cannot be replaced for *all* possible Q_2s (equation 1) because, for example, composition of Q_1' and Q_2' does not satisfy the required property (since Q_2' does not provide the user airline preference required by Q_1'). Typically, to identify a candidate replacement of Q_1 for all possible Q_2s, it is required that the candidate exhibits more functionality than Q_1. However, if the property φ is considered, the condition of substitutability can be relaxed. For example, Q_1'' (Figure 1(c)) can act as a replacement for Q_1 for all possible Q_2s since φ is satisfied in all the cases. It is worth mentioning that Q_1 and Q_1'' are not functionally/observationally/simulation equivalent.

[3]We rely on translators similar to the ones proposed in [17] to translate existing Web service specifications (e.g., BPEL) to their corresponding LTS representations.

3 Modeling Web Services & Properties

We describe services using *labeled transition systems* (LTS) [14] and properties using *mu-calculus* [8] formulae. LTS and mu-calculus are expressive enough to represent any event-driven systems (such as Web services) and temporal logic properties (e.g., CTL, LTL, CTL*), respectively. The choice of LTS and mu-calculus formalisms in our setting enables us to draw on a large body of existing results in our approach to address the service substitutability problem.

3.1 Labeled Transition System

An LTS is defined by the tuple $Q = (S, s_0, A, \Delta)$ where S is the set of states, $s_0 \in S$ is the start state and A is the set of actions of the form $\{m?, m!, m, \tau\}$. An action of the form $m?$ denotes an input, $m!$ denotes an output, m is an atomic action, and τ is an internal action. The transition relation is $\Delta \subseteq S \times A \times S$. We will write $s \xrightarrow{a} s'$ to denote the relation $(s, a, s') \in \Delta$. We will say that an action a is inverse of action b, denoted by $inv(a, b)$, if and only if $a = m?$ and $b = m!$, or vice versa.

Definition 1 (LTS composition) *Given* $Q_1 = (S_1, s_{0,1}, A_1, \Delta_1)$ *and* $Q_2 = (S_2, s_{0,2}, A_2, \Delta_2)$*, their composition*

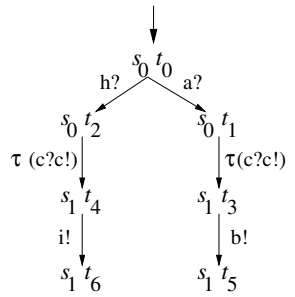


Figure 2: Composition of Q_1 and Q_2 ($Q_1 \mid\mid Q_2$)

under a set of "restrictions" R, denoted by $(Q_1 \mid\mid Q_2)\backslash R$, is a tuple $Q = (S, s_0, A, \Delta)$ where $S \subseteq S_1 \times S_2$, $s_0 = (s_{01}, s_{02})$, and $A \subseteq A_1 \cup A_2 \cup \{\tau\}$. The transition relation Δ is defined as:

1. $(s_1, s_2) \xrightarrow{\tau} (t_1, t_2)$ *if there exists* $s_1 \xrightarrow{a} t_1$, $s_2 \xrightarrow{b} t_2$, $\text{inv}(a, b)$ *and* $a, b \in R$

2. $(s_1, s_2) \xrightarrow{a} (t_1, t_2)$ *if* $a \notin R$ *and there exists (i)* $s_1 \xrightarrow{a} t_1$, $s_2 = t_2$, *or (ii)* $s_2 \xrightarrow{a} t_2$, $s_1 = t_1$

In the above, the restriction R defines the set of actions on which Q_1 and Q_2 must make synchronized moves and generate a τ-transition in the composition. This is similar in flavor to CCS processes (whose semantics is given in terms of LTS) and their composition [14]. In Figure 1, the services synchronize only on input/output actions `c?`, `c!`, `d?`, `d!`, `e?` and `e!` since rest of the actions are input/output action from/to the client. Figure 2 presents the composition of LTSs Q_1 and Q_2.

3.2 Mu-Calculus

Mu-Calculus is an expressive logic with explicit least and greatest fixed point operators for representing temporal properties. It is more general than logics like LTL (linear temporal logic), CTL (computation tree logic), CTL*—properties expressed in these logics can be represented using mu-calculus.

The syntax of mu-calculus formulas is defined over a set of fixed point variables $\mathcal{X}$ and actions A as follows:

$$\phi \rightarrow tt \mid ff \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a]\phi \mid X \mid \sigma X.\phi$$

where $a \in A$, $X \in \mathcal{X}$ and $\sigma \in \{\mu, \nu\}$. The $\langle . \rangle$ and $[.]$ are modal operators referred to as diamond and box modalities, respectively. The operator μ is the least fixed point operator while ν is greatest fixed point operator. The formula of the form $\sigma X.\varphi$ is a fixed point formula where X is said to be *bound* by the fixed point operator σ. We will consider formulas that contain only bound variables. We will write $def(X) = \sigma X.\varphi$ for $\sigma X.\varphi$.

The semantics of mu-calculus formula φ, denoted by $[\![\varphi]\!]_e$ is given by the set of states of an LTS $M = (S, s_0, A, \Delta)$ which satisfies the formula. Here e is a mapping of the form $e : \mathcal{X} \rightarrow 2^S$. Figure 3 presents the semantics of mu-calculus formulas using M and e. In the figure,

1.	$[\![tt]\!]_e$	=	S
2.	$[\![ff]\!]_e$	=	$\emptyset$
3.	$[\![X]\!]_e$	=	$e(X)$
4.	$[\![\varphi_1 \wedge \varphi_2]\!]_e$	=	$[\![\varphi_1]\!]_e \cap [\![\varphi_2]\!]_e$
5.	$[\![\varphi_1 \vee \varphi_2]\!]_e$	=	$[\![\varphi_1]\!]_e \cup [\![\varphi_2]\!]_e$
6.	$[\![\langle a \rangle \varphi]\!]_e$	=	$\{s \mid \exists s \xrightarrow{a} s' \wedge s' \in [\![\varphi]\!]_e\}$
7.	$[\![[a]\varphi]\!]_e$	=	$\{s \mid \forall s \xrightarrow{a} s' \Rightarrow s' \in [\![\varphi]\!]_e\}$
8.	$[\![\mu X.\varphi]\!]_e$	=	$f^n_{X,e}(\emptyset)$
9.	$[\![\nu X.\varphi]\!]_e$	=	$f^n_{X,e}(S)$

Figure 3: Semantics of mu-calculus formula

the propositional constant *tt* is a satisfied by all states while *ff* is not satisfied by any state. The semantics of conjunctive and disjunctive formula expressions are the intersection and the union of the semantics of the conjuncts and disjuncts, respectively. $\langle a \rangle \varphi$ is satisfied by states which have at least one a-successor that satisfies φ. The dual $[a]\varphi$ is satisfied by the states whose all a-successors satisfy φ. The semantics of fixed point variable X is defined by the mapping function e. Finally, semantics of least and greatest fixed point formula expressions are defined using the function $f_{X,e}(\hat{S}) = [\![\varphi]\!]_{e[X \mapsto \hat{S}]}$ where $def(X) = \sigma X.\varphi$ and $\hat{S} \subseteq S$. Here, $e[X \mapsto S']$ denotes an update to the mapping function such that $e[X \mapsto S'](Y) = S'$ if $X = Y$ and $e(Y)$ otherwise. It can be immediately shown that $f_{X,e} : 2^S \rightarrow 2^S$ is monotonic over the lattice of subsets of state-set S, i.e. for all $S_1 \subseteq S_2 \subseteq S$: $f_{X,e}(S_1) \subseteq f_{X,e}(S_2)$. Following Tarski-Knaster theorem [20], the fixed point semantics as n applications of the function $f_{X,e}$ where $n = |S|$. The semantic-computation of least fixed point starts from the bottom of the subset-lattice $\emptyset$ while that of the greatest fixed point proceeds from the top element in the lattice S. We will use the above semantic definition in the subsequent sections.

We say that an LTS $M = (S, s_0, A, \Delta)$ satisfies a fixed point formula φ ($M \models \varphi$) if and only if $s_0 \in [\![\varphi]\!]_e$. Note that, if φ only contains bounded fixed point variables then its semantics is independent of e. We will use $s \in [\![\varphi]\!]$ and $s \models \varphi$ interchangeably.

Example 1 *Consider the LTS Q_2 shown in Figure 1(d) where s_0 is the start state. We want to verify whether the $M \models \varphi$ where φ is defined as $\mu X.\langle c! \rangle \text{tt} \vee \langle - \rangle X$. We use "-" as a short-hand to "any" action. The semantic computation proceeds as follows:*

$$\begin{aligned} f_{X,e}(\emptyset) &= [\![\langle c! \rangle \text{tt} \vee \langle - \rangle X]\!]_{e[X \mapsto \emptyset]} = \{s_0\} \\ f^2_{X,e}(\emptyset) &= f_{X,e}(f_{X,e}(\emptyset)) = [\![\langle c! \rangle \text{tt} \vee \langle - \rangle X]\!]_{e[X \mapsto f_{X,e}(\emptyset)]} \\ &= [\![\langle c! \rangle \text{tt} \vee \langle - \rangle X]\!]_{e[X \mapsto \{s_0\}]} = \{s_0\} \end{aligned}$$

The computation can be terminated as fixed point is reached and the semantics is $\{s_0\}$. The formula is satisfied by states which eventually reach a state that has an "c!" transition. Therefore, $M \models \varphi$ as $s_0 \models \varphi$.

The preceding example contains one fixed point variable, although in general multiple fixed point variables may appear in a formula resulting in a *nested* fixed point formula. The nesting depth of the formula is defined by the number of nestings of fixed point formula expressions present in the formula. We will use $nd(\varphi)$ to denote nesting depth of the formula φ.

Going back to the example in Figure 1, composition of Q_1 and Q_2 realizes the functionality or the property where *after the client sends an input message for searching flight reservations (a?), the composite service eventually provides an output message (b!) with the list of available options*. No other input/output is demanded/provided from/to the client. We will refer to the required functionality as φ which can be represented as:

$$\langle \texttt{a?} \rangle \mu X.\, (\langle \texttt{b!} \rangle tt \vee \langle \tau \rangle X) \tag{3}$$

The formula represents the behavior where an `a?` action is followed by a `b!` action after finitely many τ steps, where `a?` corresponds to flight input information to be used for search and `b!` corresponds to the search results.

4 Quotienting Mu-Calculus Properties

We now proceed to describe the quotienting of a mu-calculus property (or formula) against an LTS. Given a formula φ and an LTS Q, quotienting (φ / Q) results in a formula ψ which must be satisfied by the environment of Q such that the overall composition satisfies φ. Quotienting of φ against Q is equivalent to the quotienting of φ against s_0, the start state of Q. Such techniques have been used to solve problems in (a) model checking ring protocols [1], (b) verification of parameterized systems [21] and (c) controller synthesis of discrete event systems [3]. Each of these techniques define quotienting on the basis of the definition of composition of two components and with respect to the specific domain being considered.

We will define the quotienting function $(\varphi/_{T,R} s)$ as $/ : \Phi \times S \times \mathcal{R} \times \mathcal{T} \rightarrow \Phi$ where $\varphi \in \Phi$, $s \in S$ of an LTS Q, $R \in \mathcal{R}$ is the restricted action set (the actions on which Q must synchronize with its environment) and $T \in \mathcal{T}$ is a tag set. The tag set contains elements of the form X_i^s where X is a fixed point variable in φ, $s \in S$ and i is an integer. The tag set is necessary to ensure termination of the recursive quotienting. The result of $(\varphi/_{T,R} s)$ is another mu-calculus formula that must be satisfied by the *environment* state t such that $(s,t) \models \varphi$ under the restriction R.

Figure 4 presents the quotienting function. Each rule follows from the semantics of mu-calculus formula expression described in Figure 3. Rule 1 states that any environment state when composed with s can satisfy *tt* while Rule 2 states that there is no environment state that can be composed with s to satisfy *ff*.

Rules 3 and 4 follow from the fact that semantics of conjunctive and disjunctive formulas are intersection and union of the semantics of conjuncts and disjuncts, respectively.

Rule 5 handles quotienting of diamond modal formula

1. $(tt/_{T,R} s) = tt$
2. $(ff/_{T,R} s) = ff$.
3. $(\varphi_1 \wedge \varphi_2/_{T,R} s) = (\varphi_1/_{T,R} s) \wedge (\varphi_2/_{T,R} s)$.
4. $(\varphi_1 \vee \varphi_2/_{T,R} s) = (\varphi_1/_{T,R} s) \vee (\varphi_2/_{T,R} s)$.
5. $$(\langle a \rangle \varphi/_{T,R} s) = \langle a \rangle (\varphi/_{T,R} s) \vee \begin{cases} \left(\bigvee_{s':s \xrightarrow{c} s'} \langle b \rangle (\varphi/_T s')\right) & \text{if } a = \tau \wedge \exists s' : s \xrightarrow{c} s' \\ & \quad \wedge\, inv(b,c) \ \wedge\ b,c \in R \\ ff & \text{otherwise} \end{cases} \vee \begin{cases} \left(\bigvee_{s':s \xrightarrow{a} s'} (\varphi/_T s')\right) & \text{if } \exists s' : s \xrightarrow{a} s' \wedge\ a \notin R \\ ff & \text{otherwise} \end{cases}$$
6. $$([a]\varphi/_{T,R} s) = [a](\varphi/_{T,R} s) \wedge \begin{cases} \left(\bigwedge_{s':s \xrightarrow{c} s'} [b](\varphi/_T s')\right) & \text{if } a = \tau \wedge \exists s' : s \xrightarrow{c} s' \\ & \quad \wedge\, inv(b,c) \ \wedge\ b,c \in R \\ tt & \text{otherwise} \end{cases} \wedge \begin{cases} \left(\bigwedge_{s':s \xrightarrow{a} s'} (\varphi/_T s')\right) & \text{if } \exists s' : s \xrightarrow{a} s' \wedge\ a \notin R \\ tt & \text{otherwise} \end{cases}$$
7. $$(\sigma X.\varphi_x/_{T,R} s) = \begin{cases} \sigma X_i^s.(\varphi_x/_{T \cup \{X_i^s\},R} s) \text{ if } X_i^s \notin T \\ \sigma X_{i+1}^s.(\varphi_x/_{T[X_i^s/X_{i+1}^s],R} s) \\ \quad \text{otherwise} \end{cases}$$
8. $$(X/_{T,R} s) = \begin{cases} X_i^s \text{ if } X_i^s \in T \\ (\sigma X.\varphi_x/_{T,R} s) \text{ otherwise} \\ \quad \text{where } def(X) = \sigma X.\varphi_x \end{cases}$$

Figure 4: Quotienting Rules

expressions. There are three possible cases by which (s,t), where t is the environment state composed with s, can satisfy $\langle a \rangle \varphi$. Each case leads a separate disjunct in the result of quotienting:

- t can make a move on a to t' such that (s,t') satisfies φ. This is represented by the first disjunct where the environment state (in this case t) is left with the obligation to satisfy the diamond modality $\langle a \rangle$ and at least one its a-successor must satisfy the result of $(\varphi/_{T,R} s)$.
- The second case corresponds to the case when $a = \tau$ and there exists transitions from s and t on which they can synchronize and move to s' and t', respectively, such that (s',t') satisfies φ. This case represents the situation when both s and t makes a synchronous move. As such the second disjunct in quotienting imposes on the environment to satisfy at least one diamond modal obligation $\langle b \rangle$ when s has a c-successor and b and c are inverse of each other. Further, b and c must be present in the restricted set.
- Finally, the state s can satisfy the diamond obligation $\langle a \rangle$. This case corresponds to the situation when s makes a move on a while t remains idle.

Note that quotienting automatically handles the possible

non-determinism at the state s by considering disjunction over the all the relevant outgoing transitions. The Rule 6 is the dual of Rule 5 and can be similarly explained.

Rules 7 and 8 represent the quotienting of fixed point formula expressions and fixed point formula variables. The rules closely follow the fixed point semantics as presented in Section 3.2. Consider $(\sigma X.\varphi/_{_{T,R}} s)$. Recall that (s,t) belongs to the semantics of $\sigma X.\varphi$ if it belongs to the semantics of φ. Quotienting $\sigma X.\varphi$ results in a new formula over fixed point variable X_i^s (case 1 of Rule 7). The new variable X_i^s is added to the tag set T. Case 2 in Rule 7 states that if X_i^s is already present in the tag set denoting that $\sigma X.\varphi$ has already been quotiented against s (i times), then a new formula variable X_{i+1}^s is used and the tag set is appropriately updated; $T[X_i^s/X_{i+1}^s]$ means that X_i^s is replaced by X_{i+1}^s in T.

The new formula generated from quotienting φ against s may lead to quotienting X against s. The situation corresponds to the case where $(s,t) \in [\![\sigma X.\varphi]\!]_e$ when $(s,t) \in e(X)$. As such, quotienting of X against s is equal to X_i^s (the last fixed point variable resulting from quotienting of $\sigma X.\varphi$ against s). This is shown in Rule 8, case 1. On the other hand, if quotienting φ against s leads to quotienting X against s' where s' has not be used to quotient $\sigma X.\varphi$ before, the situation corresponds to the case where $(s,t) \in [\![\sigma X.\varphi]\!]_e$ when $(s',t') \in e(X)$. Furthermore, since s' has not been used to quotient $\sigma X.\varphi$, it implies that $(s',t') \in e(X)$ can only occur if $(s,t) \in f^k_{X,e}(\hat{S})$ and $(s',t') \in f^{k-1}_{X,e}(\hat{S})$. This leads to case 2 in Rule 8 where X is replaced by its definition and quotiented against the state (s' in the above example case) under consideration.

It can be proved that the number of times a fixed point expression in φ is quotiented by a state has an upper bound of $|S|^{nd(\varphi)}$, where $nd(\varphi)$ is the nesting depth of formula φ. For $nd(\varphi) = 1$, the proof of the above statement is trivial since for a formula expression of the form $\sigma X.\varphi_x$, φ is quotiented at most $|S|$ times (see Rules 7 and 8).

For the general case, let $g(n)$ be the number of times a fixed point expression in φ is quotiented by any state when $nd(\varphi) = n$. Now, let us construct a new formula expression $\sigma Z.\varphi_z$ such that φ is a subformula of φ_z and Z is a subformula in φ. That is, the nesting depth of $\sigma Z.\varphi_z$ is $n+1$ and $\sigma Z.\varphi_z$ can be quotiented by every state in the LTS such that, for each such quotienting operation, the inner formula φ will be quotiented $g(n)$ times (induction hypothesis). Therefore, the total number of times a formula in $\sigma Z.\varphi_z$ is quotiented against any state is $|S| \times g(n)$, i.e. $g(n+1) = |S| \times g(n)$. Proceeding further, $\forall i \geq 1.g(i) = |S|^i$.

The following theorem which asserts the soundness and completeness of the quotienting rules follows from the above discussion.

Theorem 1 (Soundness & Completeness of Quotenting Rules) *Given Q_1 = (S_1, s_{01}, A_1, Δ_1), Q_2 = (S_2, s_{02}, A_2, Δ_2), restriction set R and a mu-calculus formula φ, the following holds:*

$$((Q_1 \mid\mid Q_2)\backslash R \models \varphi) \Leftrightarrow (Q_2 \models (\varphi/_{_{\emptyset,R}} Q_1))$$

$(\varphi/_{_{\emptyset,R}} Q_1)$
$= (\langle a?\rangle \mu X.(\langle b!\rangle tt \vee \langle \tau\rangle X))/t_0$
$= (\langle a?\rangle \mu X_1^{t_0}.(\langle b!\rangle tt \vee \langle \tau\rangle X_1^{t_0})) \vee \varphi_{X_1^{t_1}}$ Rules 5, 1

$\varphi_{X_1^{t_1}}$
$= \mu X_1^{t_1}.(\langle b!\rangle tt \vee \langle c!\rangle \varphi_{X_1^{t_3}} \vee \langle d!\rangle \varphi_{X_1^{t_3}} \vee \langle \tau\rangle X_1^{t_1})$ Rules 7, 5, 4, 1

$\varphi_{X_1^{t_3}}$
$= \mu X_1^{t_3}.(tt)$ Rules 7, 5, 1

(i)

$(\varphi/_{_{\emptyset,R}} Q_1'')$
$= (\langle a?\rangle \mu X.(\langle b!\rangle tt \vee \langle \tau\rangle X))/t_0$
$= (\langle a?\rangle \mu X_1^{t_0}.(\langle b!\rangle tt \vee \langle \tau\rangle X_1^{t_0})) \vee \varphi_{X_1^{t_1}}$ Rules 5, 1

$\varphi_{X_1^{t_1}}$
$= \mu X_1^{t_1}.(\langle b!\rangle tt \vee \langle c!\rangle \varphi_{X_1^{t_3}} \vee \langle d!\rangle \varphi_{X_1^{t_3}} \vee \langle e!\rangle \varphi_{X_1^{t_3}} \vee \langle \tau\rangle X_1^{t_1})$ Rules 7, 5, 4, 1

$\varphi_{X_1^{t_3}}$
$= \mu X_1^{t_3}.(tt)$ Rules 7, 5, 1

(ii)

Figure 5: Results of quotienting φ (equation 3 in section 3) by: (i) Q_1 (Figure 1(a)) and (ii) Q_1'' (Figure 1(c))

Example 2 *Consider the sample services Q_1 and Q_1'' in Figure 1(a, c) and the mu-calculus formula φ in Equation 3. The goal is to verify whether Q_1'' can substitute Q_1 for all possible Q_2s in Figure 1(d, e, f) in the context of φ. The results, $\psi = (\varphi/_{_{\emptyset,R}} Q_1)$ and $\psi'' = (\varphi/_{_{\emptyset,R}} Q_1'')$, are shown in Figure 5 where $R = \{c?, c!, d?, d!\}$.*

4.1 Complexity of Quotienting

The complexity of quotienting operation can be derived from the size of the result of the quotient. Given a formula φ and the set of states S against which it is quotiented, the number of times each subformula in φ is quotiented by each state in S is $|S|^{nd(\varphi)}$ (see above) which is also the nesting depth of the resultant quotient. Next, observe that the Rules 5 and 6 considers all (*matching*) outgoing transitions of the participating state and generates modal obligation following the transitions. As such, the size of the quotient is amplified by a factor of B where B is the maximum branching factor of the LTS. The overall size of the quotient is $O(|\varphi| \times |S|^{nd(\varphi)} \times B)$, where $|\varphi|$ is the size of φ.

5 Substitutability of Web Services

We now proceed to show that the environment-independent and environment-dependent variants of the service substitutability problem introduced in section 1 (equations 1 and 2) can be reduced to mu-calculus *satisfiability* using the notion of quotienting presented above.

5.1 Environment-Independent Substitutability

In this case, the problem is to determine whether Q_1' can replace Q_1 for *all* possible environments Q_2s for which

$Q_1 \mid\mid Q_2 \models \varphi$ (Equation 1).

From Theorem 1, the property to be satisfied by all Q_2s, such that $Q_1 \mid\mid Q_2 \models \varphi$, is $(\varphi /_{\emptyset,R} Q_1)$. Similarly, the obligation on possible environments of Q_1' (say Q_2's) is $(\varphi /_{\emptyset,R} Q_1')$.

If the set of Q_2s is a subset of Q_2's, then the following holds:

$$\begin{array}{lll} \forall Q_2 : (Q_1 \mid\mid Q_2)\backslash R \models \varphi & \Leftrightarrow & Q_2 \models (\varphi /_{\emptyset,R} Q_1) \\ & \Rightarrow & Q_2 \models (\varphi /_{\emptyset,R} Q_1') \\ & \Leftrightarrow & (Q_1' \mid\mid Q_2)\backslash R \models \varphi \end{array}$$

The environment-independent substitutability is, therefore, reduced to satisfiability of $(\varphi /_{\emptyset,R} Q_1) \Rightarrow (\varphi /_{\emptyset,R} Q_1')$.

5.2 Environment-Dependent Substitutability

Assume that the composition of services Q_1 and Q_2 under the restriction R realizes the functionality described by the mu-calculus formula φ: $(Q_1 \mid\mid Q_2)\backslash R \models \varphi$. In the event it is required to replace Q_1 by Q_1', it suffices to verify whether Q_1' satisfies $(\varphi /_{\emptyset,R} Q_2)$. The verification of $Q_1' \models (\varphi /_{\emptyset,R} Q_2)$ can be done using mu-calculus model checkers which takes as input the mu-calculus formula, LTS and returns true or false depending on whether the LTS satisfies the formula or not (using semantics of mu-calculus as described in Section 3.2). If the Q_1' satisfies $(\varphi /_{\emptyset,R} Q_2)$, it follows from Theorem 1 that $(Q_1' \mid\mid Q_2)\backslash R \models \varphi$. Therefore, Q_1' can replace Q_1 in the environment in which Q_1 is composed with Q_2 to satisfy φ (Equation 2).

5.3 Mu-calculus Satisfiability

Satisfiability of mu-calculus formula is performed by reducing the problem to emptiness problem of alternating tree automata [9] or identifying the winning strategy in a parity game [2, 3, 18]. Details of the technique are beyond the scope of this paper. At a high-level, these techniques determines the satisfiability of mu-calculus formula on the basis of satisfiability of its subformulas and take special care to handle fixed point satisfiability. The complexity of satisfiability checking, is therefore, exponential to the number of subformulas of the formula under consideration.

5.4 Complexity of Substitutability

Recall that, quotienting φ against an LTS containing set of states S results in a formula (say ψ) of size $O(|\varphi| \times |S|^{nd(\varphi)} \times B)$ and nesting depth $|S|^{nd(\varphi)}$ (see Section 4.1). Complexity of satisfiability of ψ is exponential to the number of subformulas in ψ. Note that at each nesting depth in ψ the number of subformulas is $O(|\varphi| \times B)$; therefore the complexity for satisfiability checking is $O(|S|^{nd} \times 2^{|\varphi| \times B})$.

For Q_1, φ, determining whether Q_1' can replace Q_1 in an environment independent fashion in the context of φ, has the complexity $O(\mathtt{max}(|S_1|^{nd(\varphi)}, |S_1'|^{nd(\varphi)}) \times 2^{|\varphi| \times \mathtt{max}(B_1, B_1')})$ where S_1, B_1 and S_1', B_1' are set of states and maximum branching factors of Q_1 and Q_1', respectively.

	Environment-Dependent	*Environment-Independent*
Context-Dependent	This paper	This paper, [4]
Context-Independent	[4, 6]	[4–6, 10–13, 19]

Figure 6: Classification of Web Service Substitutability

6 Related Work

Compatibility and substitutability of Web services has received significant attention in the literature and we classify a representative set of such work into different categories in Figure 6. Bordeaux et al. [6] introduce three different notions of compatibility of Web services (namely, observation indistinguishability, unspecified receptions, and deadlock freeness) and use them as basis to define context-dependent and independent substitution of Web services, which are described using labeled transition systems. [11] extends [6] to handle non-determinism in service behaviors. Mecella et al. [13] also introduce a formal model for substitutability of Web services that are represented using state machines. Substitution is considered in the context of a composition where replaceability analysis is done based on trace equivalence [14]. The idea of simulation equivalence is applied by Benatallah et al. [4] for determining compatibility and substitutability of Web services. The authors define different notions of compatibility and replaceability of Web services (and their interaction protocols) and provide various operators (e.g., intersection, projection) to determine replacement of services.

On a somewhat different note, Beyer et al. [5] provide three different languages (namely, signatures, consistency interfaces and protocol interfaces) for specifying Web service interfaces, and consider subsumption equivalence and subsumption ordering to ascertain replaceability of services. Martens et al. [12] devise an approach for determining behavioral and syntactical compatibility between Web services that are modeled as petri nets. Here also, the authors adopt trace equivalence and bisimulation equivalence to determine similarity between two petri net models. However, their approach also ensures that there are no deadlocks between the compatible processes. Taher et al. [19] also determine similarity between two services based on the interface descriptions (specified in WSDL). This work assumes the formation of communities of services that provide similar functionality, and hence can be substituted by analyzing syntactical and semantical similarity between service descriptions. Another interesting approach is reported by Li and Jagadish [10] where the authors adopt graph-matching techniques for substitutability analysis. Their work represents service interfaces using graphs, where nodes and edges correspond to states and transitions of a service respectively, and applies graph-similarity heuristics to determine compatibility between services.

The focus of this paper is on two variants of context-specific service substitutability, namely environment-dependent and environment-independent service substi-

tutability. In light of the results presented in section 5, the (context-specific) simulation equivalence requirement explored in [4, 5, 12] corresponds to checking the satisfiability of $(\varphi/_{\emptyset,R} Q_1) \Leftrightarrow (\varphi/_{\emptyset,R} Q_1')$. This will force Q_1 and Q_1' to be behaviorally identical with respect to the context φ and will correspond to a stronger requirement for substitutability.

7 Summary and Discussion

Determining substitutability of a service with another is an important problem in service-oriented computing. In this paper, we focus on the problem of context-specific service substitution which requires that some desired property φ of the component being replaced is maintained despite its substitution by another component. We introduce two variants of the context-specific service substitutability problem, namely, environment-dependent and environment-independent substitutability that relax the requirements for substitutability relative to simulation or observational equivalence between services. The proposed solution to these two problems is based on the well-studied notion of "quotienting" which is used to identify the obligation of the environment of a service being replaced in a specific context. We demonstrate that both environment-dependent and environment-independent service substitutability problems can be reduced to quotienting of φ against the service being replaced and the replacement service and hence, to satisfiability of the corresponding mu-calculus formulae. The correctness of our technique follows from the correctness of the individual steps of quotienting and satisfiability.

In the current setting, we did not take into consideration the data parameters, i.e. messages being exchanged by the services. In our prior work, [15, 16] we have discussed equivalence between services where the communication paradigm includes messages that can potentially have an infinite domain. We plan to explore the application of quotienting-based approach to context-specific substitutability to the setting of message-based communication. In particular, we plan to investigate the possible use of value-passing LTS/CCS and more powerful value-passing mu-calculus [21]. Furthermore, we assumed synchronous communication between the services. Hence, consideration of services which communicate asynchronously [7] and analysis for substitutability in such a setting are of interest. In addition to the above, we plan to work on adopting this work into a more dynamic setting where services can be replaced at runtime by automatic re-composition that takes into consideration not only the functional, but also the non-functional requirements for substitution. Finally, an interesting aspect that deserves further research is analyzing failure of substitution, i.e., what action can be taken when an existing service cannot be replaced by another.

Acknowledgment. This research has been supported in part by the Iowa State University Center for Computational Intelligence, Learning & Discovery (`http://www.cild.iastate.edu`), NSF grant 0509340 to Samik Basu and NSF-ITR grant 0219699 to Vasant Honavar.

References

[1] H. Andersen. Partial Model Checking (extended abstract). In *Logic in Computer Science*, 1995.

[2] A. Arnold, A. Vincent, and I. Walukiewicz. Games for Synthesis of Controllers with Partial Observation. *Theoretical Computer Science*, pages 7–34, 2003.

[3] S. Basu and R. Kumar. Quotient-based Control Synthesis for Non-Deterministic Plants with Mu-Calculus Specifications. In *45th IEEE Conference on Decision and Control*, 2006.

[4] B. Benatallah, F. Casati, and F. Toumani. Representing, Analysing and Managing Web Service Protocols. *Data and Knowledge Engineering*, 58(3):327–357, 2006.

[5] D. Beyer, A. Chakrabarti, and T. Henzinger. Web Services Interfaces. In *15th World Wide Web Conference*, pages 148–159. ACM Press, 2005.

[6] L. Bordeaux, G. Salaün, D. Berardi, and M. Mecella. When are Two Web Services Compatible? In *5th International Workshop on Technologies for E-Services*, pages 15–28. LNCS 3324, Springer-Verlag, 2004.

[7] T. Bultan, J. Su, and X. Fu. Analyzing Conversations of Web Services. *IEEE Internet Computing*, 10(1):18–25, 2006.

[8] E. A. Emerson. Model Checking and the Mu-Calculus. In *Symposium on Descriptive Complexity and Finite Model*, pages 185–214. American Mathematical Society Press, 1997.

[9] E. A. Emerson and C. S. Jutla. The Complexity of Tree Automata and Logics of Programs. *SIAM Journal of Computing*, 29(1):132–158, 1999.

[10] Y. Li and H. Jagadish. Compatibility Determination in Web Services. In *ICEC Workshop on Workshop on E-Government and Web Services*, 2003.

[11] F. Liu, L. Zhang, Y. Shi, L. Lin, and B. Shi. Formal Analysis of Compatibility of Web Services via CCS. In *1st International Conference on Next Generation Web Services Practices*, pages 143–148. IEEE Computer Society, 2005.

[12] A. Martens, S. Moser, A. Gerhardt, and K. Funk. Analyzing Compatibility of BPEL Processes. In *International Conference on Internet and Web Applications and Services*, pages 147–155. IEEE CS Press, 2006.

[13] M. Mecella, B. Pernici, and P. Craca. Compatibility of e-Services in a Cooperative Multi-platform Environment. In *1st International Workshop on Technologies for e-Services*, pages 44–57. LNCS 2193, 2001.

[14] R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.

[15] J. Pathak, S. Basu, and V. Honavar. Modeling Web Services by Iterative Reformulation of Functional and Non-Functional Requirements. In *4th International Conference on Service Oriented Computing*, pages 314–326. LNCS 4294, Springer-Verlag, 2006.

[16] J. Pathak, S. Basu, R. Lutz, and V. Honavar. Parallel Web Service Composition in MoSCoE: A Choreography-based Approach. In *4th IEEE European Conference on Web Services*, pages 3–12. IEEE CS Press, 2006.

[17] M. Pistore, P. Traverso, P. Bertoli, and A. Marconi. Automated Synthesis of Composite BPEL4WS Web Services. In *3rd Intl. Conference on Web Services*, pages 293–301. IEEE Press, 2005.

[18] C. Stirling. Games and Modal Mu-Calculus. In *Second International Workshop Tools and Algorithms for Construction and Analysis of Systems*, pages 298–312, 1996.

[19] Y. Taher, D. Benslimane, M.-C. Fauvet, and Z. Maamar. Towards an Approach for Web services Substitution. In *10th Intl. Database Engineering and Applications Symposium*, pages 166–173. IEEE CS Press, 2006.

[20] A. Tarski. A Lattice-Theoretical Fixpoint Theorem and Its Applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[21] P. Yang, S. Basu, and C. Ramakrishnan. Parameterized Verification of Pi-Calculus Systems. In *12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 42–57. LNCS 3920, Springer-Verlag, 2006.