A Finite Representation of all Substitutable Services and its Applications

Jarungjit Parnjai[1,⋆], Christian Stahl[12,⋆⋆], and Karsten Wolf[3,⋆⋆⋆]

[1] Humboldt-Universität zu Berlin, Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany
{parnjai, stahl}@informatik.hu-berlin.de
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[3] Universität Rostock, Institut für Informatik
18051 Rostock, Germany
karsten.wolf@uni-rostock.de

Abstract. We present a finite representation of all *substitutable* services P' of a given service P. We show that our approach can be used for at least two applications: (1) given a finite set of services $\mathcal{P} = \{P_1, ..., P_n\}$, we provide a representation of all services P' that can substitute every $P_i \in \mathcal{P}$, and (2) given a service P'' that cannot substitute a service P, we find the most similar service P^* to P'' that can substitute P.

1 Introduction

The paradigm of Service-Oriented Computing (SOC)[1] uses a service as a building block for designing flexible business processes by means of service composition. The behavior of a service is subject to changes. Driven by the cost and time to meet the deadline, a new version of a service is hardly reconstructed from scratch. Instead, a service will be *substituted* by a new version, which can be derived by updating its current functionality or adding in a new functionality.

We consider a service P to be *substitutable* by a service P' if P' cooperates deadlock-freely with every partner that P cooperates deadlock-freely with. That is, substituting P by P' *preserves* every deadlock-free cooperating partner of P.

In this paper, we present an *operating guideline* approach to represent the set of all substitutable services. This representation is helpful for at least two applications. Given a finite set of services $\mathcal{P} = \{P_1, ..., P_n\}$, we show that a finite representation of all services P' that can substitute every $P_i \in \mathcal{P}$ can be computed with the help of operating guidelines [2, 3] of services. Furthermore, we show that errors in a non-substitutable service can be corrected automatically using a simulation-based graph edit distance as introduced in [4].

⋆ Funded by the DFG-Graduiertenkolleg 1324 “METRIK”.
⋆⋆ Funded by the DFG project “Substitutability of Services” (RE 834/16-1).
⋆⋆⋆ Supported by the DFG project “Operating Guidelines for Services” (WO 1466/8-1).

The remainder of this paper is organized as followed. Section 2 recalls some formalisms and substitutability notion. Section 3 presents a finite representation of all services P' that can substitute a given service P. Section 4 outlines a method to compute a finite representation of all services P' that can substitute all services $P_i \in \{P_1, ..., P_n\}$ that are given. Section 5 shows how to correct errors in a non-substitutable service P'' with respect to a given service P. Finally, Section 6 concludes the paper.

2 Background

We model the behavior of a service P with a *service automaton*. A service automaton is a finite state automaton with a set Q of states, a set $F \subseteq Q$ of final states, an initial state $q_0 \in Q$, a set I of input interfaces, a set O of output interfaces (I and O are pairwise disjoint), and a *non-deterministic* transition relation $\delta \subseteq Q \times \{I \cup O \cup \{\tau\}\} \times Q$. The edges are labeled with output message $x \in O$ sent to (labeled "$!x$") the environment, or input message $x \in I$ received from (labeled "$?x$") the environment, or internal move (label "τ"). A non-final state with no outgoing transition is called a *deadlock*.

Given two service automata P and R, their composition $P \oplus R$ is a service automaton in which its set of states is the cartesian product of Q_P, Q_R, and the set of all multisets of pending messages between P and R. We assume that two composable service automata have compatible interfaces ($I_P = O_R$ and $I_R = O_P$), but all other constituents are pairwise disjoint. The composition $P \oplus R$ is deadlock-free if $P \oplus R$ does not contain a deadlock. R is a *strategy* of P iff $P \oplus R$ is deadlock-free. The strategy relation is symmetric, that is, R is a strategy of P implies P is also a strategy of R. We write $\mathit{Strat}(P)$ to denote the set of all strategies of P. See [2] for further details.

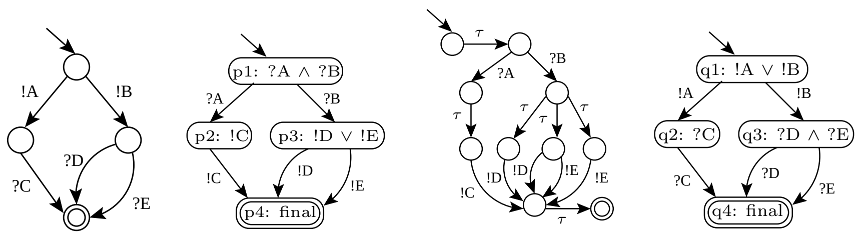
Throughout this paper, we assume a deadlock-free composition, i.e., there always exists at least one strategy for a given service.

An operating guideline $OG(P)$ of P is a deterministic service automaton S^{ϕ} where each state q of S is annotated with a Boolean formula $\phi(q)$. A matching relation between states of a service automaton and S^{ϕ} are used to characterize a set of service automata. We write $\mathit{Match}(S^{\phi})$ to denote the set of all service automata S' that satisfies a matching relation with S^{ϕ}. $OG(P)$ characterizes the (possibly infinite) set of all strategies of P, i.e., $\mathit{Match}(OG(P)) = \mathit{Strat}(P)$ [2].

Figure 1(a) depicts a service automaton P_1 and Fig. 1(b) depicts an operating guideline of P_1.

We define our substitutability notion called *accordance*. A service P' substitutes a service P under accordance (P' accords with P) iff every strategy of P is also a strategy of P', i.e., $\mathit{Strat}(P) \subseteq \mathit{Strat}(P')$. We assume that P and P' are *interface equivalent* ($I_P = I_{P'}$ and $O_P = O_{P'}$) and write $\mathit{Accord}(P)$ to denote the (possibly infinite) set of all services P' that substitute P under accordance.

[3] presents an algorithm to decide whether P' substitutes P under accordance using their operating guidelines.



(a) P_1 (b) $OG(P_1)$ (c) $MS(P_1)$ (d) $OG(MS(P_1))$

Fig. 1. (a) service automaton P_1, (b) an operating guideline of P_1, (c) MS of P_1, and (d) an operating guideline of MS of P_1.

3 Representing All Substitutable Services

Given a service P we show how to calculate an operating guideline that represents the set $Accord(P)$ of all services P' that can substitute P under accordance.

Definition 1 (Maximal Strategy, *MS*). *Let P be a service automaton and $OG(P) = S^{\phi}$ be its operating guideline. A* maximal strategy *of P, denoted $MS(P)$, is obtained from S by replacing every node q by a non-deterministic internal choice between all the valid combinations of outgoing edges from q w.r.t. satisfying assignment in $\phi(q)$.*

Figure 1(c) depicts a maximal strategy $MS(P_1)$ of P_1 (in Fig. 1(a)) and Fig. 1(d) depicts its operating guideline $OG(MS(P_1))$.

$MS(P_1)$ is obtained from the underlying service automaton of $OG(P_1)$ according to Definition 1. For example, the node p_3 of the underlying service automaton of $OG(P_1)$ is replaced by the non-deterministic τ choice between three valid combinations of outgoing transitions that satisfy assignment in $\phi(p_3) = !D \vee !E$ in $OG(P_1)$. These three combinations are (1) a transition labeled $!D$, (2) a transition labeled $!E$, and (3) two transitions labeled $!D$ and $!E$. Clearly, $MS(P_1)$ is a strategy of a service P_1. That is, $MS(P_1) \in Match(OG(P_1))$.

Mooij and Voorhoeve [5] have proven that for a strategy R of P, each strategy of $MS(P)$ is also a strategy of R.

Proposition 1 ([5]). *Let P be a service automaton such that $Match(OG(P)) \neq \emptyset$. Then for all $R \in Match(OG(P))$ holds: $Strat(MS(P)) \subseteq Strat(R)$.*

By the help of Proposition 1 we prove that $OG(MS(P))$ represents the set $Accord(P)$ of all service automata P' that can substitute a service automaton P under accordance.

Theorem 1 (Characterizing all substitutable services). *Let P and P' be two service automata. Let $OG(P)$ be an operating guideline of P. Then, P' substitutes P under accordance iff $P' \in Match(OG(MS(P)))$.*

Proof. We will show that $\mathit{Accord}(P) = \mathit{Match}(OG(MS(P)))$.

Consider $\mathit{Accord}(P) = \{P' \mid \mathit{Strat}(P) \subseteq \mathit{Strat}(P')\}$. Since the strategy relation is a symmetric relation, we conclude that $\mathit{Accord}(P) = \{P' \mid \forall R \in \mathit{Strat}(P) : P' \in \mathit{Strat}(R)\} = \bigcap_{R \in \mathit{Strat}(P))} \mathit{Strat}(R)$. Since $\mathit{Match}(OG(P)) = \mathit{Strat}(P)$, $\mathit{Accord}(P) = \bigcap_{R \in \mathit{Match}(OG(P))} \mathit{Strat}(R)$ follows.

Next, we will show that $\bigcap_{R \in \mathit{Match}(OG(P))} \mathit{Strat}(R) = \mathit{Strat}(MS(P))$. We know $MS(P) \in \mathit{Strat}(P)$ and $\mathit{Strat}(P) = \mathit{Match}(OG(P))$. Therefore, we can conclude that $\bigcap_{R \in \mathit{Match}(OG(P))} \mathit{Strat}(R) \subseteq \mathit{Strat}(MS(P))$. Proposition 1 asserts that for all $R \in \mathit{Match}(OG(P))$ holds: $\mathit{Strat}(MS(P)) \subseteq \mathit{Strat}(R)$. Therefore, we can conclude that $\bigcap_{R \in \mathit{Match}(OG(P))} \mathit{Strat}(R) \supseteq \mathit{Strat}(MS(P))$. Consequently, $\bigcap_{R \in \mathit{Match}(OG(P))} \mathit{Strat}(R) = \mathit{Strat}(MS(P))$ immediately follows.

We know $\mathit{Strat}(MS(P)) = \mathit{Match}(OG(MS(P)))$. Thus, we can conclude that $\bigcap_{R \in \mathit{Match}(OG(P))} \mathit{Strat}(R) = \mathit{Match}(OG(MS(P)))$.

Consequently, $\mathit{Accord}(P) = \mathit{Match}(OG(MS(P)))$. $\square$

Theorem 1 shows that the operating guideline $OG(MS(P))$ is a finite representation of all P' that can substitute P under accordance.

Our result enables a service designer to effectively derive P' from $OG(MS(P))$. Clearly, P' can substitute P under accordance, as it matches with $OG(MS(P))$. The designer can also use P' as a template to tailor a new version P'' by filling P' with some internal actions. This way, it can be decided if P'' substitutes P under accordance by checking if $P'' \in \mathit{Match}(OG(MS(P)))$.

With our approach, we can also decide accordance (as presented in [3]) of two services P'' and P by checking if $P'' \in \mathit{Match}(OG(MS(P)))$.

4 Conjoining Substitutable Services

Suppose a service designer would like to design a new service which can support all potential customers of both a hotel booking service and a flight booking service. The representation of all services that can substitute both booking services is helpful for the designer. With this representation, the designer can decide whether such a new service does exist, and in case it does, a well-suited upgrade of a new service can be derived immediately from such a representation.

For a finite set $\mathcal{P} = \{P_1, .., P_n\}$ of service automata, we show that the intersection $\bigcap_{P_i \in \mathcal{P}} \mathit{Accord}(P_i)$ of sets of all services that accord with every P_i can be represented by the product of all operating guidelines of maximal strategy $MS(P_i)$ of P_i, where $P_i \in \mathcal{P}$.

The product of two operating guidelines [3] is defined as an operating guideline that characterizes the intersection of all service automata that match with these two operating guidelines. The product of two operating guidelines assumes that both operating guidelines are interface equivalent.

Proposition 2 ([3]). *Let $OG_{\otimes} = OG(S_1) \otimes OG(S_2)$ be the product of operating guidelines $OG(S_1)$ and $OG(S_2)$, Then, $\mathit{Match}(OG_{\otimes}) = \mathit{Match}(OG(S_1)) \cap \mathit{Match}(OG(S_2))$.*

Corollary 1 (Characterizing intersection of substitutable services). *Let P_1 and P_2 be two service automata. Let $OG(P_1)$ be an operating guideline of P_1 and $OG(P_2)$ be an operating guideline of P_2. Then,*

$$Match(OG(MS(P_1)) \otimes OG(MS(P_2))) = Accord(P_1) \cap Accord(P_2).$$

Proof. Follows from Proposition 2 and Theorem 1. □

Corollary 1 shows that we can use the product of operating guidelines to compute the finite representation of all services that accords with both P_1 and P_2. In case the returned product describes an empty set, there is no service automaton P' that can substitute both P_1 and P_2 under accordance.

Since the product $\otimes$ of operating guidelines is commutative and associative, the result from Corollary 1 can be easily generalized to the product of any finite number n of operating guidelines of $MS(P_i)$, where $P_i \in \{P_1, .., P_n\}$.

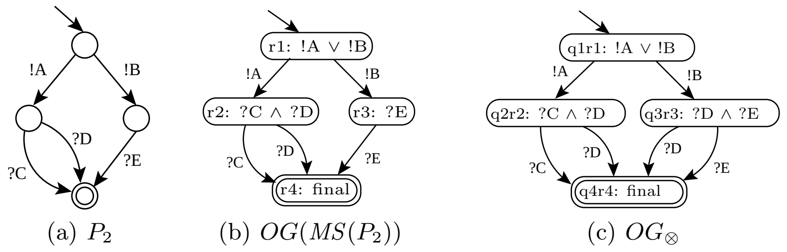


Fig. 2. (a) service automaton P_2, (b) an operating guideline of MS of P_2, and (c) the product $OG_\otimes$ of $OG(MS(P_1))$ and $OG(MS(P_2))$.

Figure 2(c) depicts $OG_\otimes$ as a finite representation of all services that can substitute both P_1 (Fig. 1(a)) and P_2 (Fig. 2(a)) under accordance. $OG_\otimes$ is the synchronous product of $OG(MS(P_1))$ and $OG(MS(P_2))$, where each node is annotated with the conjunction of the two Boolean formulas of the corresponding states of $OG(MS(P_1))$ and $OG(MS(P_2))$. For example, the node $q2r2$ in $OG_\otimes$ is annotated with $\phi(q2r2) = ?C \wedge ?D$, which is the conjunction of $\phi(q2) = ?C$ in $OG(MS(P_1))$ and $\phi(r2) = ?C \wedge ?D$ in $OG(MS(P_2))$.

5 Correcting Non-Substitutable Services

Suppose a service designer has designed an ill-suited upgrade of a travel agency service that does not accord with the travel agency service. Synthesizing a new well-suited upgrade of the service using an approach proposed in Section 3 may not be sufficient, as the well-suited upgrade might be very different and totally ignore the structure of its ill-suited upgrade version. The designer might prefer to reuse an ill-suited upgrade of the service instead of synthesizing a new well-suited upgrade of the service.

To reuse an ill-suited upgrade of the service, the errors found in the ill-suited upgrade can be fixed manually. Nevertheless, the manual correction is a tedious and error-prone procedure. This scenario motivates a method to synthesize a well-suited upgrade of the service automatically from its ill-suited upgrade.

Given two service automata P and P'' where P'' does not accord with P, we propose a procedure to correct P'' with respect to P. The errors in P'' can be detected and corrected automatically using a simulation-based graph edit distance, as introduced in [4] to fix a faulty service to cooperate deadlock-freely in a choreography. The approach takes P'' and $OG(MS(P))$ as its input, computes the most similar service automaton P^* to P'' such that $P^* \in Match(OG(MS(P)))$, and returns the edit actions that are necessary to transform P'' into P^*. Clearly, P^* can substitute P under accordance, as it matches with $OG(MS(P))$. That is, P^* cooperates deadlock-freely with every strategy of P. This way, P'' can be reused, as P^* is most similar to P'', yet accords with P.

So far the simulation-based graph edit distance approach is applicable only for acyclic and deterministic services [4].

6 Conclusion

We have proposed an approach to characterize the set $Accord(P)$ of all services P' that can substitute a service P under accordance. We have shown that a finite representation of $Accord(P)$ can be computed using the concept of a maximal strategy [5] and its operating guideline [2]. With this representation, we can decide accordance of two services and derive from it a new service that accords with a given service.

We have shown two applications of our approach. Given a finite set of services $\mathcal{P} = \{P_1, ..., P_n\}$, we provide a representation of the intersection of $Accord(P_i)$ for all $P_i \in \mathcal{P}$ with the help of the product of operating guidelines [3]. For a service P'' that cannot substitute a service P, we provide an automatic correction procedure to transform P'' into the most similar P^* such that P^* accords with P with the help of the simulation-based graph edit distance [4].

References

1. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson - Prentice Hall, Essex (2007)
2. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In Kleijn, J., Yakovlev, A., eds.: ICATPN 2007. Volume 4546 of LNCS., Springer-Verlag (2007) 321–341
3. Stahl, C., Massuthe, P., Bretschneider, J.: Deciding substitutability of services with operating guidelines. LNCS ToPNoC **II**(5460) (2008) 172–191
4. Lohmann, N.: Correcting deadlocking service choreographies using a simulation-based graph edit distance. In Dumas, M., Reichert, M., eds.: BPM 2008. Volume 5240 of LNCS., Springer-Verlag (2008) 132–147
5. Mooij, A.J., Voorhoeve, M.: Proof techniques for adapter generation. In Bruni, R., Wolf, K., eds.: WS-FM 2008 Milan, Italy, Proc. LNCS, Springer-Verlag (2008)