Automated Choreographer Synthesis for Web Services Composition Using I/O Automata

Saayan Mitra\(^1\) \quad Ratnesh Kumar\(^1\) \quad Samik Basu\(^2\)

\(^1\) Department of Electrical and Computer Engineering
\(^2\) Department of Computer Science
Iowa State University, Ames, IA
Email:{saayan, rkumar, sbasu}@iastate.edu

Abstract

We study the problem of synthesis of a choreographer in Web service composition for a given set of services and a goal. Services and goal are represented using i/o automata which can succinctly and precisely describe the interfaces of the services. Our technique considers existence and synthesis of two types of the choreographers: a simple choreographer capable of only relaying outputs from one service to input of another and a transducing choreographer which is capable of storing and reusing inputs/outputs from the services. The central theme of our technique relies on generating i/o automata representation of all possible choreographed behavior of existing services (captured in form of universal service automaton, a concept introduced in this paper) and verifying that the goal can be simulated by the universal set of choreographed behaviors.

1 Introduction

Service oriented solutions have made major inroads in the computing world with their impact in the domain of e-service, e-commerce, e-business etc. Web services form one of the central themes in service oriented architectures. Web services can be viewed as functions or sequences of functions which provide outputs for specified inputs. An interesting problem in Web services is to address the question of whether (and how) multiple, heterogeneously developed services can be employed in a co-operative fashion to realize and develop new desired services. The problem is referred to as Web service composition.

Service composition relies on specifications of input-output behavior of services which are typically represented using service description languages, e.g., WS-BPEL, OWL-S, WSDL [14]. At a high-level, composition amounts to identifying how the output of one can be connected to the input of the other and realize the appropriate output for a specific input.

There are two main aspects of composition problems. First, as the services are developed independently, the description of input-output of one service may be different from another and hinder in connecting the output of one to the input of the other. For example, one service may require an input of temperature in Celsius while the service recording the temperature outputs the temperature in Fahrenheit. Such incompatibilities are referred to as semantic mismatch of services. The second issue is closely related to the functionality, i.e., services need to be composed in a specific fashion to realize a desired or new service. We will refer to the new service as goal. The solution to the problem may require identifying a choreographer which supervises the behavior of services in such a way that the goal is achieved.

Driving Problem. In this paper, we investigate automata-theoretic approach to address the second problem: given a set of services and a goal, is it possible to generate a choreographer which can communicate with the services to realize the goal? In our technique, we consider existence and synthesis of two types of choreographers: simple choreographer and transducing choreographer. Simple choreographer is only capable of relaying output from one service to input of another, i.e., choreographer does not buffer any output for later use. On the other hand, transducing choreographer is capable of storing already seen inputs and outputs and using them to provide inputs to services at a later

\(^*\)This work is supported in part by NSF grants NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, NSF-ECS-0424048, NSF-ECS-0601570, and NSF 0509340.
The choreographer does not exist. A choreographer is possible for the said goal; otherwise, possible composed behavior, we infer that synthesizing a choreographer. If the goal behavior is simulated by all realized from the services using a simple or transducing choreographer. The phenomenon is referred to as hiding. In short, the transducing choreographers can store, use and ignore inputs and outputs to and from services as and when necessary.

Our Solution. Given a set of services, our solution relies on identifying all possible behaviors that can be realized from the services using a simple or transducing choreographer. If the goal behavior is simulated by all possible composed behavior, we infer that synthesizing a choreographer is possible for the said goal; otherwise, choreographer does not exist.

The contributions of this paper are summarized as follows:

1. Service composition is concerned with the input-output behavior of the services. We use i/o automata to precisely represent this behavioral aspect.

2. Existence of two types of choreographers are considered. We present a sequence of steps which uses i/o automata representation of services and develops a new set of automata representing all possible choreographed behavior of the services. If the generated automaton simulates the goal (also represented as i/o automaton), we say that a (simple or transducing) choreographer can be synthesized.

3. The steps for synthesizing a choreographer systematically shows how a simple choreographer synthesis problem can be extended to solve the transducing choreographer synthesis problem. The solution methodology provides better understanding of the composition problem and paves way for future advancements where more complex choreographers can be considered.

4. The proposed technique is sound and complete.

Organization. The rest of the paper is organized as follows. Section 2 describes an example which will be used in the rest of the paper to explain the salient features of our technique. Section 3 provides the relevant background and describes i/o automata. Sections 4 and 5 present the techniques to verify existence of simple and transducing choreographers respectively. Related work is discussed in Section 6 followed by the conclusion in Section 7.

2 Illustrative Example

Consider the sequence diagrams in Figure 1: (a) and (b) which show the input-output behavior of two existing services. The sequence of activities can be obtained by top-down scanning of the corresponding diagram. $S_1$ takes as input social security number (SSN) and produces as output the credit card number (CCN); while $S_2$ takes as input CCN twice and outputs credit approval (Appr) and loan approval (Loan). The inputs to the services (when functioning on their own) come from their environment and outputs are provided to the environment. By environment, we mean the client or user who is using the services.

Assume that the developer wants to create a new service (referred to as goal service) $S_G$ which takes as input SSN from the client and outputs Loan (Figure 1(c)). The existing services $S_1$ and $S_2$ do not provide the required $S_G$. However, an intermediary or a choreographer can be synthesized which relays and controls information between $S_1$ and $S_2$ and replicates the input-output behavior of $S_G$. Figure 1(d) shows the choreographed $S_1$ and $S_2$. Note that the choreographer acts as the environment of the services; it relays the SSN input from the client to $S_1$ and stores the CCN output from $S_1$. It does not relay CCN to the client (it hides this output), instead the choreographer provides CCN as input to $S_2$ to obtain the outputs Appr.

![Figure 1](image1.png)

![Figure 2](image2.png)
Example 1 is the set of inputs, the set of states, \( x \) and \( \to \) to the transition, is the origin state of the transition, of \( T \times X \times A \) goal. be synthesized given a set of existing services and a goal.

3 Preliminaries

3.1 Services as Automata

We describe the (goal and component) services as input/output automata whose states represent the configuration of the services and interstate transitions define the way in which the services evolve from one configuration to another. Each transition is enabled in the presence of a specific input and results in a specific output. Formally, input/output automaton is defined as follows:

Definition 1 (I/O Automaton) An i/o automaton \( A \) is defined by a tuple \((X, X^0, I, O, T)\), where, \( X \) is the set of states, \( X^0 \subseteq X \) is the set of initial states, \( I \) is the set of inputs, \( O \) is the set of outputs and \( T \subseteq X \times I \times O \times X \) is the set of transitions. An element of \( T \) represented by \( t = (x, i, o, x') \) is such that \( x \in X \) is the origin state of the transition, \( i \in I \) is the input to the transition, \( o \in O \) is the output of the transition and \( x' \in X \) is the destination state of the transition.

We will often use \( x \xrightarrow{i/o} x' \) to denote \((x, i, o, x') \in T\).

Example 1 Consider the automata representation of services (Figure 1(a,b)) in Figure 2. The input-labels of each transition in Figure 2 are the inputs to the services from their corresponding environment, while the output-labels are the outputs provided by the service following an input. The sequencing in the service automata can be obtained from the sequence of input-output in its sequence diagram. WLOG we assume that the input and output always alternate\(^1\).

3.2 Role of a Choreographer

Recall that the problem of service composition is to identify whether a set of existing services can be used to realize a specified goal (a new service). The goal can be represented by an i/o automaton and an existing service is said to realize the goal if all possible behaviors of the goal is also present in the service.

Typically however, a goal cannot be realized from one service. For example, consider the goal automaton in Figure 3. It cannot be realized from any of the services in Figure 2(a,b). In such cases, a choreographer is required to control the services such that their composition with the choreographer realizes the specified goal. In our setting there are two important features of such a composition: (a) a service does not communicate with the client or other services directly and (b) the choreographer cannot produce any input on its own; instead it relies on services or the client for the inputs.

The main idea behind choreographer synthesis is to realize each goal-transition by executing certain service-transitions in a certain sequence, and relaying the inputs/outputs of the service-transitions in accordance to that sequence. The input to the first service-transition in the sequence is the same as the goal-transition input, the output from the last service-transition in the sequence is the same as goal-transition output, and the output from any other service-transition is the same as the input to the next service-transition in the sequence. Such a goal-transition is said to belong to the closure of the sequence of service-transitions. We can also project such a sequence of service-transitions on individual services to obtain a set of sequences, each one of which belongs to a single service. We then say that the goal-transition is realizable from that set of sequences.

The set of all possible service-transition sequences are first obtained by taking the interleaving product of the service-automata. A “closure” operation is then performed over the sequences of service-transitions (obtained through interleaving) to identify the goal-transitions that a certain sequence of service-transitions is able to realize. In the following we formalize these notions.

Before proceeding to formally define a choreographer, we present the notational convention and background definitions. We will use \( \alpha, \beta, \mu, \nu \) and their subscripted versions to define sequences. Concatenation will be denoted by \( . \) (dot). For any two strings \( \alpha \) and \( \beta \), \( \alpha \preceq \beta \) denotes that \( \alpha \) is a prefix of \( \beta \).

Definition 2 (Interleaving product of sequences)
The interleaving product of sequences \( (\|) \) is defined inductively as: \( \epsilon \| \alpha = \alpha \| \epsilon = \alpha; \)

\(^1\) A sequence of inputs followed by outputs can be easily represented if transition labels are sequence-based.
\[ (i/o.\mu) \parallel (i'/o'.\nu) = i/o.(\mu||i'/o'.\nu) + i'/o'(i/o.\mu \parallel \nu). \]

**Definition 3 (Closure)** The closure of a string \( \alpha = i_1/o_1.i_2/o_2.\ldots i_n/o_n \in (I \times O)^* \), denoted by \( \text{CL}(\alpha) \), is equal to
\[
\left\{ i_1/o_1.i_2/o_2.\ldots i_m/o_m.\ldots i_n/o_n \mid \exists m < k : \forall j, m \leq j < k : i_j = o_j \right\}
\]

For example,
\[
\text{CL}(a/b.b/c.c/d, a/c.c/d, a/d, a/b.b/d) = \{a/b.b/c.c/d, a/c.c/d, a/d, a/b.b/d\}
\]

**Definition 4 (Realizability)** A given \( \alpha \in (I \times O)^* \) is said to be realizable from \( \{\beta_n \mid \beta_n \in (I_n \times O_n)^*\} \), where \( I \subseteq \bigcup_n I_n \) and \( O \subseteq \bigcup_n O_n \), if and only if \( \exists \mu, \nu \in \bigcup_{1 \leq n \leq N} \beta_n : \alpha \in \text{CL}(\mu) \), where \( \text{CL}(\mu) \) denotes interleaving product.

Let \( \alpha = a/d, \beta_1 = a/b.c/d \text{ and } \beta_2 = b/c. \) Then \( \beta_1 \parallel \beta_2 = \{a/b.c/d.b/c\} \) and there exists a sequence \( \mu = a/b.b/c/c/d \) in the above set, the closure of which includes \( a/d \) (see example after Equation 1). As \( \alpha \in \text{CL}(\mu) \), \( \alpha \) is said to be realizable from \( \{\beta_1, \beta_2\} \).

Using the definition of closure and realizability, we next define the notion of a choreographer, which maps a sequence belonging to a goal to a set of sequences, one for each available service. The mapping must be causal (i.e., prefix-preserving) for it to be implementable and also must satisfy the requirement of realizability.

**Definition 5 (Choreographer)** Given a goal automaton \( G = (X_g, X_g^0, I_g, O_g, T_g) \) and \( N \) service automata of the form \( A_n = (X_n, X_n^0, I_n, O_n, T_n) \) for \( n \leq N \) such that \( I_g \subseteq \bigcup_n I_n \) and \( O_g \subseteq \bigcup_n O_n \), a choreographer is a function of the form \( C : (I_g \times O_g)^* \rightarrow \bigcap_{n \leq N} (I_n \times O_n)^* \) such that

1. \( \forall \mu, \nu \in (I_g \times O_g)^* : \mu \preceq \nu \Rightarrow \forall n \leq N : C_n(\mu) \preceq C_n(\nu) \)
2. \( \forall \mu \in (I_g \times O_g)^* : \mu \text{ is realizable from } \{C_n(\mu)\} \)

We present below the automata-theoretic approach to verify the existence of a choreographer for a given set of the services and a specified goal.

### 4 Existence of Choreographer

We first consider the case where the choreographer is capable of only relaying output from one service to input of another. It does not store and/or reuse any inputs from the client or outputs from the existing services. The functionality is denoted by the closure defined in Equation 1.

To identify the existence of such a choreographer, we first define an interleaving product automaton from the existing service automata. This automaton captures all possible interleaved behavior of the participating service automata.

**Definition 6 (Interleaving Product)** Given a set of service automata \( A_1, A_2, \ldots, A_N \) where \( A_n = (X_n, X_n^0, I_n, O_n, T_n) \) for \( 1 \leq n \leq N \), the interleaving product of \( \{A_n\} \) is defined as an automaton \( \|A_n = (X, X^0, I, O, T) \), where \( X \subseteq X_1 \times X_2 \times \ldots \times X_N \) is the set of states, \( X^0 = X_1^0 \times X_2^0 \times \ldots X_N^0 \) is the set of initial states, \( I = \bigcup_{1 \leq n \leq N} I_n \) is the set of inputs and \( O = \bigcup_{1 \leq n \leq N} O_n \) is the set of outputs. Finally, \( T \subseteq X \times I \times O \times X \) is the set of transitions such that

\[ (x_1, \ldots, x_n) \xrightarrow{i/o} (x'_1, \ldots, x'_n) \in T \Leftrightarrow \exists i \leq N : x_i \xrightarrow{i/o} x'_i \in T_i \land \forall j \leq N, j \neq i : x'_j = x_j \]

**Example 2** Figure 4(a) shows the automaton \( \|A_n \) obtained from service automata \( A_1 \) and \( A_2 \) in Figures 2(a) and (b). Each state in the \( \|A_n \) is annotated with the tuple \( (i, j) \) where \( i \) represents the state of \( A_1 \) and \( j \) represents the state of \( A_2 \).

The \( \|A_n \) represents the set of behaviors that can be realized from the interleaving of services \( A_n \)'s. To identify the sequences that can be obtained through some choreography, we define the operation closure of an automaton, which follows from the Equation 1.

**Definition 7 (Closure of I/O Automaton)** Given an \( i/o \) automaton \( A = (X, X^0, I, O, T) \), the closure of \( A \), denoted by \( A^* \), is defined as \( (X, X^0, I, O, T^*) \) where

\[
x_{i_1/o_1} x_{i_2/o_2} x_{i_3/o_3} \ldots \in T^* \Leftrightarrow \exists k : \begin{cases} x_{i_1/o_1} x_{i_2/o_2} x_{i_3/o_3} \in T \land \\ x_{i_k/o_k} x' \in T \land \\
\forall 1 < j \leq k : i_j = o_{j-1} \end{cases}
\]

The universal service automaton, denoted \( U \), is defined as the closure of the service-product automaton, i.e., \( U := (\|A_n) \)

Note the above definition of closure ensures that for all \( \mu \in A, CL(\mu) \subseteq A^* \).

**Example 3** Going back to the example services \( A_1 \) and \( A_2 \) in Figures 2(a) and (b), the corresponding \( U \)-automaton obtained from \( \|A_n \) of Figure 4(a) is shown...
in Figure 4(b). Observe that the U-automaton also includes transitions that can be realized when a choreographer relays output of one service to the input of the other. For example, the transition \((1, 1) \xrightarrow{\text{CCN/Appr}} (2, 2)\) is generated from the transitions \((1, 1) \xrightarrow{\text{CCN/Appr}} (2, 1)\) and \((2, 1) \xrightarrow{\text{CCN/Appr}} (2, 2)\). It captures the situation when a choreographer sends the SSN input (from client) to A1 at state 1 and relays the CCN output from A1 to the input of A2 at its state 1. As a result, both A1 and A2 move from their corresponding states 1 to 2.

U-automaton contains all possible choreographed behavior of the services. Proceeding further, the goal automaton \(A_G\) is realizable from the services using some choreographer if and only if all possible behavior of \(A_G\) is simulated by U-automaton.

**Definition 8 (Simulation [13])** Given an i/o automaton \(A = (X, X^0, I, O, T)\), for all \(x_1\) and \(x_2\) in \(X\), \(x_1\) is simulated by \(x_2\) if they are related by the largest simulation relation denoted by \(x_1 \sqsubseteq x_2\) and defined as:

\[
x_1 \sqsubseteq x_2 \iff [\forall x_1' : x_1 \xrightarrow{\alpha} x_1' \Rightarrow (\exists x_2' : x_2 \xrightarrow{\alpha} x_2')]
\]

An i/o automaton \(A_1 = (X_1, X^0_1, I_1, O_1, T_1)\) is said to be simulated by \(A_2 = (X_2, X^0_2, I_2, O_2, T_2)\), denoted by \(A_1 \sqsubseteq A_2\), if all states in \(X^0_1\) is simulated by some state in \(X^0_2\).

**Theorem 1** Given a goal \(A_G\) and a set of services \(A_1, A_2, \ldots, A_N\), the goal can be realizable from the composition of \(A_n\)s with a choreographer if and only if \(A_G \sqsubseteq U\) where \(U\) is the closure of the \(\|\)\(A_n\) obtained from \(\{A_n\}\).

**Example 4** Note that the goal in Figure 3 is not simulated by U-automaton and therefore, there exists no choreographer which can realize \(A_G\) from \(A_1\) and \(A_2\). If the requirement of a goal was to generate an Appr output for SSN input, it can be simulated by U-automaton in Figure 4(b).

5 **Transducing Choreographer**

In the previous section, we considered a choreographer which is capable of relaying output from one service to the input of another. However, it is not capable of storing the inputs and outputs. The stored information can be used to provide inputs to services at any time. In this section, we consider the choreographer that is capable of such functionality. The added functionality allows choreographer to perform inducing and hiding of actions. Inducing refers to the case where the choreographer uses some stored information to supply required input to a service while hiding refers to the case where the choreographer absorbs outputs from the services and does not provide it to the client (hidden from the client).

We will refer to such a choreographer as transducing choreographer. To identify the behavior that can be realized using a transducing choreographer, we will extend the definition of closure of a sequence (see Equation 1) in turn which will enhance the definition of realizability (Definition 4) and empower the choreographer.

Note that, sequence now has a context consisting of the history that led a choreographer to see this sequence. The history of a sequence keeps track of the set of inputs and outputs that can be used by the choreographer to transduce the said sequence.

**Definition 9 (Transduced-Closure)** Given a sequence \(\alpha = i_1/o_1, i_2/o_2 \ldots i_n/o_n \in (I \times O)^*\), its transduced closure, denoted by \(CL^T(\alpha)\), is equal to

\[
\{ \exists m < k : \forall j, m \leq j < k : i_{j+1} \in \{i_l, o_l | l \leq j\} \}
\]
For example for $\mathcal{L}^T(a/b,a/c) = \{a/b,a/c, a/c\}$. Using the notion of transduced-closure, we next define the notion of transduced-realizability.

**Definition 10 (Transduced-Realizability)**

$\alpha \in (I \times O)^*$ is said to be transduced-realizable from \{\(\beta_n \in (I_n \times O_n)^*\) if $\exists \mu \in \|n \leq N \beta_n : \alpha \in \mathcal{L}^T(\mu)\$.

A transducing choreographer that uses inducing and hiding (w.r.t. inputs and outputs seen in past) besides “chaining” is defined as follows.

**Definition 11 (Transducing-Choreographer)**

Given a goal automaton $G = (X_g, X_g^0, I_g, O_g, T_g)$ and $N$ service automata of the form $A_n = (X_n, X_n^0, I_n, O_n, T_n)$ for $1 \leq n \leq N$ such that $I_g \subseteq \bigcup_n I_n$ and $O_g \subseteq \bigcup_n O_n$, a transducing choreographer is function of the form $C^T : (I_g \times O_g)^* \to \prod_n (I_n \times O_n)^*$ satisfying

1. $\forall \mu, \nu \in (I_g \times O_g)^* : \mu \preceq \nu \Rightarrow \forall n \leq N : C^T_n(\mu) \preceq C^T_n(\nu)$
2. $\forall \mu \in (I_g \times O_g)^* : \mu$ is transduced-realizable from $\{C^T_n(\mu)\}$

Transducing choreographer is more powerful than the simple choreographer as the former is capable of choreographing more functionality (see Equation 2) from the existing services than the later.

Proceeding in a fashion similar to the one followed in Section 4, we first define an interleaving product with history, which maintains a history set of the seen inputs and outputs with each state.

**Definition 12 (Interleaving Product with History)**

Given a set of service automata $A_1, A_2, \ldots, A_N$ where $A_n = (X_n, X_n^0, I_n, O_n, T_n)$ for $1 \leq n \leq N$, the interleaving product with history is defined to be the automaton $\|n A_n = (X_H, X_H^0, I, O, T_H)$ where $X_H \subseteq X_1 \times X_2 \times \cdots \times X_N \times 2^\bigcup I$ and $O = \bigcup_n O; X_H^0 = X_1^0 \times X_2^0 \times \cdots \times X_N^0 \times \{\emptyset\}$ and $T_H \subseteq X_H \times I \times O \times X_H$ such that

$$\left( x_1, \ldots, x_N, h \right) \xrightarrow{i/o} (x'_1, \ldots, x'_N, h') \in T_H \iff \exists i \leq N : x_i \xrightarrow{i/o} x'_i \in T_i \land \forall j \leq N, j \neq i : x'_j = x_j \land h' = h \cup \{i, o\} \right.$$

The $\|n A_n$-automaton is similar to $\|n A_n$-automaton with the exception that the history of i/o are tracked at each state (so these can be used for internal inducing even when there is no external input).

**Example 5** Figure 5(a) shows the $\|n A_n$-automaton generated from $A_1$ and $A_2$ in Figures 2(a,b). Every state is labeled by the states of $A_1$ and $A_2$ and also shows that inputs and outputs that a transducing choreographer can use. For example at the start state (1,1), the choreographer does not have any stored information as it has no history of interaction while if the services are at state (2,2), the choreographer contains the information on SSN, CCN, Appr obtained from its prior interactions.

The $\|n A_n$-automaton represents the information available to a transducing choreographer. To identify
the sequences that can be choreographed by chaining along with inducing and hiding the transitions available in the \((\|A_n\|)^\ast\)-automaton, we define the operation of transduced-closure.

**Definition 13 (Transduced-Closure of Automaton)**

Given an automaton with history \(A_H = (X_H, X_H^0, I, O, T_H)\), the transduced-closure of \(A_H\) is the automaton \(U^T = (X_H, X_H^0, I, O, T_H)\) where 

\[
(x, h) \xrightarrow{i/o} (x_k, h_k) \in T_H \text{ if and only if } \exists k : \begin{align*}
(x, h) &\xrightarrow{i/o} (x_1, h_1) \in T_H \\
(x_1, h_1) &\xrightarrow{i/o} (x_2, h_2) \in T_H \\
& \quad \vdots \\
(x_{k-1}, h_{k-1}) &\xrightarrow{i/o} (x_k, h_k) \in T_H \\
& \quad \forall 1 \leq j < k : i_j \in h_j
\end{align*}
\]

**Example 6** Consider the \(U^T\)-automaton (Figure 5(b)) obtained from the \(\|A_n\|\)-automaton in Figure 5(a). \(U^T\)-automaton contains transitions that can be realized from using the history information at each state. The newly added transitions are labeled with \([i/o]\). State \((1, 1)\) has a transition to state \((1, 3)\) on \([CCN/Loan]\). The transition is obtained from the transitions \((1, 1) \xrightarrow{CCN/Appr} (1, 2)\) and \((1, 2) \xrightarrow{CCN/Loan} (1, 3)\). After the first transition, both \(CCN\) and \(Appr\) are available for future input at state \((1, 2)\). The second transition from \((1, 2)\) therefore does not rely on the environment to provide its enabling input; the input can be provided by a transducing choreographer by using the \(CCN\) from the information stored at \((1, 2)\). The second transition produces the output \(Loan\). Similar transition is added from \((2, 1)\) to \((2, 3)\) on \([CCN/Loan]\), following which \((1, 1) \xrightarrow{SSN/Loan} (2, 3)\) is also added to \(U^T\)-automaton.

**Theorem 2** Given a goal \(A_G\) and a set of services \(A_1, A_2, \ldots, A_n\), the goal can be realizable from the composition of \(A_n\)’s with a transducing choreographer if and only if \(A_G \subseteq U^T\) where \(U^T\) is the transduced-closure of the \(\|A_n\|\)-automaton obtained by taking interleaving product with history of the automata \(A_n\)’s.

**Example 7** For the Definition 11, if each goal sequence is executable in the \(U^T\)-automaton, then it is inferred that the goal is realizable from the existing services using a transducing choreographer. This is verified using the simulation relation (Definition 8). In the current example the \(U^T\) automaton in Figure 5(b) simulates the goal in Figure 3. Therefore, the goal can be realizable from the existing service automata \(A_1\) and \(A_2\) (Figure 2) by using a transducing choreographer. The choreographer first relays the \(SSN\) information from the client to \(A_1\) and then uses the \(CCN\) output from \(A_1\) as input to \(A_2\) twice to get the appropriate output \(Loan\) which is then relayed to the client. The transducing choreographer is shown in Figure 6.

Once it is verified that the goal service is simulated by the universal service automaton (\(U\) or \(U^T\) as the case may be), a choreographer can be synthesized by first identifying for each goal transition a corresponding simulating transition in the universal service automaton, and next identifying the set of service transition sequences that realize it. (Note the information about the set of service transition sequences realizing a transition of the universal service automaton is embedded in its definition.)

**6 Related Work**

A number of approaches has been developed in the recent past to detect and/or synthesize choreography based composition of services. The techniques range from manual development of choreography to more rigorous automated procedures that rely on formal methods. The techniques applying formal methods to service composition are typically based on automata theory, dynamic logic and AI planning. In the following we discuss some of the representative works in this domain.

In [5] the authors describe services as automata extended with queue and allow exchange of messages between services in an asynchronous fashion. A global watchter is developed to keep track of messages being exchanged in the composition to detect whether a specified goal service is realizable. Subsequently in [6] the authors use Spin model checker [8] to verify whether a composition correctly replicates the required goal. The message passing framework is extended in [1] which uses satisfiability of propositional dynamic logic to detect the existence of a choreographer. The work is further generalized in [3] where non-determinism in the service behavior is considered. A similar approach on service composition is presented in [15], where the main objective is to create a choreographer which makes the composition behavior bisimulation equivalent to the goal. [7] introduces the concept of lookahead for choreographers to plan ahead in delegating activities, based on the approach in [2].
A number of works investigated the applicability of AI planning techniques for service composition. These works apply techniques ranging from rule based planning [12], Situation Calculus [11, 10], query planning [18] and theorem proving [17], model checking [4, 16, 19]. In essence, the techniques reduce the problem of composition to that of planning a desired execution of workflows. The underlying basis in the planning domain also rely on state transition systems with states, actions and observations. The services communicate through messaging which again emphasizes input-output behavior of services.

Similar to the existing work, we use transition system based representation of services, more specifically, we use i/o automaton to capture the input output interfaces of each service. In fact, as noted in [9] treating a Web service as an automaton comes naturally, as it is equipped with i/o capabilities and from the point of view of composability, we are interested in the i/o functionality of the automaton. One of the distinguishing aspect of our technique is that the proposed automata theoretic approach provides valuable insights to the composition problem with respect to the capabilities of the choreographer. Recall that, in our technique, the universal-service automaton (obtained as closure of the interleaving product with/without history of service automata) captures the capability of the choreographer. If the capability of a choreographer is well-defined, the corresponding existence problem can be solved using a simulation check as described in the paper. Thus, we provide a uniform solution methodology to the choreographer existence problem.

7 Conclusion

We have reduced the problem of verifying the existence of a choreographer to a simulation problem over i/o automata. The solution relies on the construction of appropriate universal-service automaton, \( U \)-automaton or \( U^2 \)-automaton, as the case may be. One of the future avenues of research is to investigate applicability of local, on-the-fly algorithms to solve the problem. Such algorithms will explore the state-space of the universal-service automaton as and when needed and will stop exploration whenever the proof of existence of choreographer is obtained.

Another direction of research involves enhancing the i/o automaton to include variables, thereby, making it capable to capture more complex behaviors more compactly. At the current setting, we have only considered the propositional variables.

References