Received August 3, 2016, accepted September 22, 2016, date of publication September 28, 2016, date of current version January 4, 2017.

Digital Object Identifier 10.1109/ACCESS.2016.2614325

A Petri Net Method for Compatibility Enforcement to Support Service Choreography

JING BI[1], (Senior Member, IEEE), HAITAO YUAN[2], (Member, IEEE), AND MENGCHU ZHOU[3,4], (Fellow, IEEE)
[1]Beijing Engineering Research Center for IoT Software and Systems, School of Software Engineering, Beijing University of Technology, Beijing 100124, China
[2]School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China
[3]Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA
[4]Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, China

Corresponding author: H. Yuan (htyuan@bjtu.edu.cn)

This work was in part supported by the Beijing Natural Science Foundation under Grant 4164090 and National Natural Science Foundation of China (NSFC) under Grant 61374148.

ABSTRACT Non-local choice mismatch is one of the most important problems in the Internet-scale and service-based workflow ecosystems. The state-of-the-art method can solve it by generating adaptors to check deadlock-freeness based on a reachability graph. The states in the reachability graph give clues to re-design the composition. Deadlocks are resolved via an iterative process. However, this method is inefficient due to the overlook of the future deadlock state and the requirement of many interactions with a developer. In this paper, an abnormity prevention strategy based on an optimal controller is proposed for collaboration services described by Web Services Choreography Description Language. To overcome the deficiencies of the previous work, we describe service choreography by using service workflow nets. Then, by combining structure and reachability analyses, we formulate a different reachability graph called controlled reduced one. We next develop a maximally permissive state feedback control policy to prevent abnormity. We finally construct an optimal controller for the administrator of service composition to avoid deadlocks in service choreography. The advantage of our methodology is verified via an example.

INDEX TERMS Service choreography, Petri net, hybrid optimal controller, deadlock prevention, Web service.

I. INTRODUCTION

Internet has helped us create the popular and most newest application and computing paradigms, such as Web 3.0, cloud computing, big data, mobile Internet, social networking, and the Internet of Things. Due to its open, dynamic, and evolving environment, Internet continues to help us develop new software technologies. These technologies should be able to evolve to effectively deal with rapid changes, and adaptable, context aware between runtime contexts and user demands [1], [2]. Future Internet research promotes a distributed-computing environment, in which an increasing number of interactions are completed through workflow service invocations. In these Internet-scale and service-based workflow ecosystems, the dynamical coordintion and automatical composition for service workflow systems are key enablers for this concept [3], [4].

Current service composition mechanisms mostly support *service orchestration*, based on Web Services Business Process Execution Language (WS-BPEL) [5], a centralized method to compose multiple services into a larger application [6]. Orchestration works well in minimal context changes and static contexts with predefined services. These assumptions are unadapted in the Internet-based workflow ecosystem, where different service workflow providers and consumers keep changing and cannot be coordinated by a centralized method.

In contrast, service interaction specification languages, e.g., Web Services Choreography Description Language (WS-CDL) [7], and Web Service Choreography Interface (WSCI) [8], all provide a decentralized method that provides a looser way to design service composition by specifying participants and message protocols between them. *Service choreographies* describe peer-to-peer message exchanges among participant services from a global perspective. In this case, they are significantly different from service orchestrations in which different partners cooperate with other services

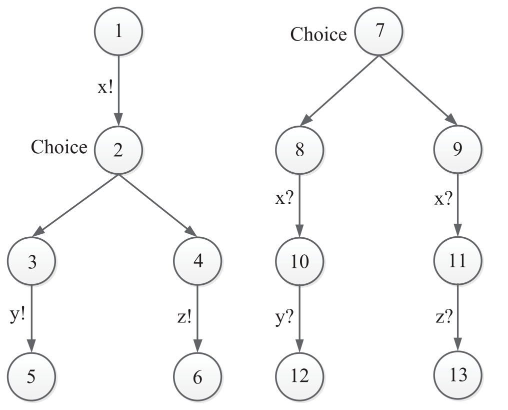


FIGURE 1. Non-local choice mismatch.

to realize an objective. However, choreography-based systems usually consider two problems: Firstly, a *realizability check* confirms if each implemented participant can realize choreography such that it conforms to the choreography role specifying its expected behavior. Secondly, a *conformance check* makes sure if the global interaction of a series of services satisfies the choreography [9]. Therefore, web service choreographies may fail to work because inconsistent semantics may be buried in their interfaces and protocols among systems of different participants.

To address these problems, researchers have proposed many methods [10]–[12]. However, most of them focus on direct composition between services. *Partial compatibility* is very common in the real-world web service interfaces and protocols. It denotes that the functionality of two or more web services is complementary, but their interfaces and protocols contain mismatches [13], [14]. Therefore, they cannot be immediately composed. Partial compatibility can cause problems including mutual waiting mismatch [15], unspecified reception mismatch [16], and non-local choice mismatch [17]–[19]. This work solves the problem of *non-local choice* mismatch for partial compatibility. This can be formulated as: the local choices of two services are different, and this leads to a discrepant behavior that causes *deadlock* [20], as shown in Fig. 1.

In Fig. 1, a service makes a local choice after it sends message x (x!). For example, it sends a message y (y!) or z (z!). However, its partner also makes a local choice, and waits for the message y (y?) or the message z (z?) after its partner receives message x (x?). If a service and its partner make the same choices, i.e., its partner is expecting to receive y but it is sending y, there is no problem. However, if its partner is expecting to receive z, it is sending y, this causes a non-local choice mismatch.

Enforcing partially compatible services to work with each other without any deadlock is valuable to the reuse of services. In the case of non-local choice mismatch, we assume that the administrator of service composition constructs an optimal enforcement controller in the message channel for service choreography, which can free programmers from identifying subtle and hard-to-identify deadlocks. Hence,

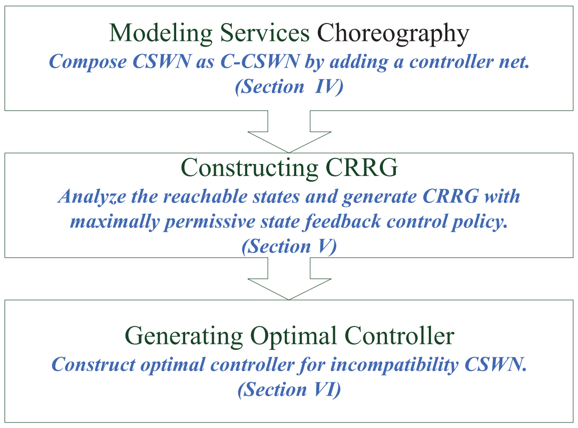

Modeling Services Choreography
Compose CSWN as C-CSWN by adding a controller net. (Section IV)
Constructing CRRG
Analyze the reachable states and generate CRRG with maximally permissive state feedback control policy. (Section V)
Generating Optimal Controller
Construct optimal controller for incompatibility CSWN. (Section VI)

FIGURE 2. Three steps in optimal enforcement of compatibility.

programmers can concentrate on business process logic and rely on the optimal controller to guarantee a *deadlock-free* execution.

As shown in Fig. 2, to address the challenge of non-local choice mismatch in service choreography, we:

1) propose a controlled composition-of-service workflow net (*C-CSWN*) model to describe service choreography;
2) construct a controlled reduced reachability graph (*CRRG*) to efficiently identify deadlocks; and
3) design an optimal controller to implement an optimal control policy for incompatible services.

Compared with the existing works, the main contributions of this paper are as follows:

1) We adopt a controlled Petri net-based approach to find out non-local choice mismatch in service choreography.
2) If non-local choice mismatch exists, we provide a solution based on an optimal controller to address incompatible service choreography.

The rest of the paper is organized as follows. Section II presents a motivating scenario for compatibility analysis. Section III proposes a controlled composition-of-service workflows net to model service choreography. Section IV carries out compatibility analysis and proposes a control policy to avoid deadlock-prone choreography. Section V gives the method to build the controller. Section VI summarizes the related work, and Section VII concludes the paper.

II. MOTIVATING SCENARIO

To illustrate the issue of non-local choice mismatch in the environment of service collaboration, the following example from supply chain domain is cited from [17]–[19] to address the question above. Assume that three web services expect to exchange messages and each has its own internal business process, i.e., Clients (**C**), Online Shop (**S**), and Third Party Checkout (**T**). They are developed by and deployed in different Internet-based partners. Creating choreography services is able to collaborate existing Internet-scale services.

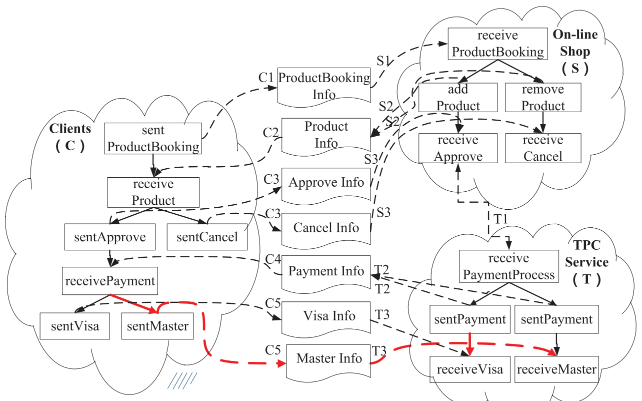


FIGURE 3. Service choreography and incompatibility in the environment of service collaboration.

For example, an online shop service provides clients with online products. It can outsource the checkout service to a third party, in which case clients are directed to a Third Party Checkout service to pay for the on-line shop service. The brief choreography workflow services of **C**, **S** and **T** are shown in Fig. 3.

C's service workflow is as follows:

C1) **C** requests **S** by submitting a booking message (*sent ProductBooking*);

C2) **C** receives a product message (*receiveProduct*) from **S**;

C3) **C** invokes its own operation of *Approve* or *Cancel* and returns the corresponding message (*sentApprove* or *sentCancel*) to **S**;

C4) If the *Approve* operation is called, the payment operation is ready, and **C** waits for a payment message (*receivePayment*) from **T**; Otherwise if *Cancel* operation is called, **C** terminates the booking workflow;

C5) **C** calls *Visa* or *Master* payment operation and returns the message (*sentVisa* or *sentMaster*) to **T**.

S's service workflow is as follows:

S1) **S** receives a booking message (*receiveProductBooking*) from **C** and enters the state of providing products;

S2) **S** calls a booking operation and returns a product message (*sentProduct*) to **C**;

S3) **S** starts the receiving operation (*receiveApprove* or *receiveCancel*) and waits for a message of *Approve* or *Cancel* from **C**.

T's service workflow is as follows:

T1) **T** receives a payment message (*receivePayment*) from **S**;

T2) **T** calls a payment operation and returns a payment message (*sentPayment*) to **C**;

T3) **T** starts the receiving operation (*receiveVisa* or *receiveMaster*) and waits for a message of *Visa* or *Master* from **C**.

The smooth interaction among **C**, **S** and **T** is essential to the success of a business transaction. In the above scenario, assume that the message interfaces of all services are completely matched, but the service operation behavior mismatches. For example, if **C** expects payment business to be with *Master* C5 (*sentMaster*), while **T** calls *Visa* T3 (*receiveVisa*) for payment, then the interaction leads to a deadlock, as shown in the thick lines of Fig. 3.

To avoid such incompatibility behavior, we propose an optimal enforcement method to identify activities that cause deadlocks and to avoid executing them, without changing the internal business process of an individual service.

III. MODELING CONTROLLER-BASED SERVICE COMPATIBILITY

A. INTERACTION PETRI NETS

Assuming that readers are familiar with Petri nets [21]–[25], we model an Interaction Service Workflow Net (*ISWN*) for a business process.

Definition 1 (ISWN): An interaction service workflow net is an extended Petri net ISWN= (P, T, F), where

1) *P is a finite set of state places. $P = P_I \cup P_M$, where*
   a) *P_I is a set of internal places*;
   b) *P_M is a set of message places; and*
   c) *$P_I \cap P_M = \emptyset$*;
2) *T is a finite set of transitions, $P \cap T = \emptyset$; and*
3) *$F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs connecting state places and transitions.*

An *ISWN* is said to be *pure* if it has no self-loops, i.e., if $\forall p \in P$, $\forall t \in T$, $[(p,t) \in F \Rightarrow (t,p) \notin F]$. If an *ISWN* is *pure*, its incidence relation can be represented by a matrix $E{:}P \times T \rightarrow \{0, 1, -1\}$, called the *incidence matrix* of the *ISWN*, where

$$E(p,t) = \begin{cases} 1 & \text{if } (t,p) \in \text{F} \\ -1 & \text{if } (p,t) \in \text{F} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

There *preset* and *postset* of a transition t are defined as ${}^{\bullet}t = \{p|(p,t) \in F\}$ and $t^{\bullet} = \{p|(t,p) \in F\}$, respectively. The *preset* and *postset* of a place p are ${}^{\bullet}p = \{t|(t,p) \in F\}$, and $p^{\bullet} = \{t|(p,t) \in F\}$, respectively.

The state of an *ISWN* is given by its current *marking* m: $P \rightarrow \{0, 1, 2, \ldots\}$ that assigns to each place of the *ISWN* a non-negative number of tokens, represented by black dots, where $m(p)$ denotes the number of tokens assigned by marking m to place p. The set of all markings defined on a net *ISWN* is M. A service workflow net system $\langle ISWN, m_0 \rangle$ is a net *ISWN* with an initial marking m_0.

A set of transitions $\tau \subseteq T$ is enabled by a marking m if

$$\forall p \in P, m(p) \geq |p^{\bullet} \cap \tau| \tag{2}$$

that is, for each place $p \in P$, $m(p)$ is greater than or equal to the number of transitions in τ where $p \in {}^{\bullet}\tau = \bigcup_{t \in \tau} {}^{\bullet}t$.

In words, firing an enabled set of transitions $\tau \subseteq T$ causes one token to be removed from each place $p \in {}^{\bullet}t$, and one token to be added to each $p \in t^{\bullet}$, for each $t \in \tau$. Let $m[\tau\rangle$ denote that τ may fire at m, and $m[\tau\rangle m'$ denote that τ may fire, resulting in m'. In much of the Petri net literature it is assumed that only a single transition can fire at any instant [22]. A *firing sequence* from a marking m_0 is a sequence of transition sets $\sigma = \tau_1 \ldots \tau_k$ such that $m_0[\tau_1\rangle m_1[\tau_2\rangle m_2 \ldots [\tau_k\rangle m_k$. We also write $m_0[\sigma\rangle$ to denote that we may fire the sequence σ at m_0, and $m_0[\sigma\rangle m_k$ to denote that the firing of σ yields m_k.

Given a service workflow net system $\langle ISWN, m_0\rangle$, the set of reachable markings is denoted $R(ISWN, m_0)$. A transition $t \in T$ is *live* if for any marking $m \in R(ISWN, m_0)$ there always exists a marking $m' \in R(ISWN, m)$ such that t is enabled by m'; a net system is live if all of the transitions are live. A transition $t \in T$ is in *deadlock* at a marking $m \in R(ISWN, m_0)$ if it cannot be enabled by any marking in $R(ISWN, m)$.

Deadlock in a Petri net model denotes the classical circular wait condition. In applications, each activity is occupying a resource required by one of the other activities, so all of the activities cannot proceed [26]. This work deals with deadlock problem in service choreography. We adopt the *state feedback control* that has been widely investigated by controlled Petri nets (*CtlPNs*) [27]–[30]. *CtlPNs* denote a class of Petri nets with external enabling conditions. These conditions are called *controlled places* that can enable an external controller to affect the progression of tokens in the net. The controller-based interaction service workflow net (*C-ISWN*) is defined based on the concept of *CtlPNs*.

Definition 2 (C-ISWN): A controller-based interaction service workflow net is defined as $ISWN^{\mathbb{C}} = (ISWN, \mathbb{C})$, where

1) *ISWN= (P, T, F) is an extended Petri net structure in Definition 1; and*
2) *$\mathbb{C} = (P_{\mathbb{C}}, T_{\mathbb{C}}, B)$ is a controller such that:*
    a) *$P_{\mathbb{C}} = \{p_{cr} | r = 1, 2\}$ is a finite set of additional control places, disjoint from P, T, $P_{\mathbb{C}} \cap P = \emptyset$;*
    b) *$B \subseteq (P_{\mathbb{C}} \times T)$ is a set of directed arcs connecting control places to transitions; and*
    c) *For a transition $t \in T$ we denote the set of input control places as $^*t = \{p_c | (p_c, t) \in B\}$, and for a control place $p_c \in P_{\mathbb{C}}$ we denote the set of output transitions as $p_c^{\bullet} = \{t | (p_c, t) \in B\}$. A transition t is said to be a controlled transition if $^*t$ is nonempty. The set of all controlled transitions is denoted by $T_{\mathbb{C}} = \{t_{cs} | s = 1, 2\}$.*

The state of a *C-ISWN* is given by its *marking*, which is the distribution of *tokens* in the state places. A set of transitions $\tau \subseteq T$ is *state enabled* under a marking m if equation (2) is satisfied.

A control for a C-CSWN is a function $u{:}P_{\mathbb{C}} \rightarrow \{0, 1\}$ associating a binary value to each controlled place. The set of all possible control policies is denoted by U. A set of transitions $\tau \subseteq T$ is said to be *control enabled* if for all $t \in \tau$, $u(p_c) = 1$ for all $p_c \in {}^*t$. Given two controls u', $u \in U$, a control $u \in U$ is said to be as *permissive* as control $u' \in U$ if $u(p_c) \geq u'(p_c)$ for all $p_c \in P_{\mathbb{C}}$. Control u is said to be *more permissive* than control u' if u is as permissive as u' and $u(p_c) > u'(p_c)$ for some $p_c \in P_{\mathbb{C}}$. It follows that $u_{zero} = 0$ is the least permissive control of all control, while $u_{one} = 1$ is the most permissive control. Note that, $u_{zero} = 0$ and $u_{one} = 1$ are abbreviated as u_{zero} and u_{one}, respectively in the later part of this paper.

Similar to the definitions in the work [39], we give the following description. Each transition is control-enabled by $u \in U$ and state-enabled by $m \in M$. Then, the collection of transition sets is denoted by $\Upsilon_e(u, m)$. Besides, any transition set $\tau \in \Upsilon_e(u, m)$ can fire. The marking changed in the net is denoted by m' that is further explained in Definition 1.

Given a marking $m \in M$ and control $u \in U$, the set of *immediately reachable markings within one state transition*, $R_1(u, m)$, is given by

$$R_1(u, m) = \{m\} \cup \{m' \in M | m' \textit{ is given for some } \tau\} \tag{3}$$

Similarly, the set of reachable markings from a marking m for a control $u \in U$, denoted by $R_\infty(u, m)$, is defined by

1) $m \in R_\infty(u, m)$;
2) $m' \in R_\infty(u, m)$, then $R_1(u, m') \subseteq R_\infty(u, m)$; and
3) all $m' \in R_\infty(u, m)$ are defined by 1) and 2).

We conclude this section by noting that the controlled service workflow net model defined is similar to the *CtlPNs* defined by Krogh [27] and Ichikawa and Hiraishi [28].

B. MODEL SERVICE COMPATIBILITY WITH CONTROLLER

The composition of web services can realize a distributed business process. There are two major ways to define the composition. One way is to specify the *orchestration* of a whole set of web services [10], [31] by a global model. The other way is to specify the *collaboration* with its partners by each web service [7], [32]. This work mainly follows the second way.

This section first shows how to model the composition of services with *ISWNs*. It is assumed that the message order and the compatibility at the syntactic level are both satisfied, i.e., the output and input interfaces have the same operation sequences and message types. Therefore, $ISWN_1$ and $ISWN_2$ are composable if they can collaborate using a set of common places. The composition of *ISWN*s (*CSWN*) is defined as follows.

Definition 3 (Composition of ISWNs): Given two ISWNs, $ISWN_i = (P_i, T_i, F_i)$, $i \in \{1, 2\}$ that are syntactically and semantically compatible workflow nets, their workflow net CSWN = $ISWN_1 \oplus ISWN_2$ is given by $(\overline{P}, \overline{T}, \overline{F})$, where

1) *$\overline{P} = P_I \cup P_M$, satisfying:*
    a) *$P_I = P_{I1} \cup P_{I2}$, $P_{I1} \cap P_{I2} = \emptyset$; and*
    b) *$P_M = P_{M1} \cup P_{M2}$, $P_A = P_{M1} \cap P_{M2} \neq \emptyset$ is a set of common message places;*
2) *$\overline{T} = T_1 \cup T_2$; and*
3) *$\overline{F} = F_1 \cup F_2$*

The composition of $ISWN_1$ and $ISWN_2$ via common message places P_A is denoted as $CSWN = ISWN_1 \oplus ISWN_2$ where $\oplus$ denotes a composition operator. Later, we also use this operator when we define composition via a controller.

As a quality criterion, the notion of *soundness* [33] is used to characterize workflow nets. Soundness needs every initiated process to reach a proper final state. In addition, each transition needs to be relevant. Therefore, there must be at least one behavior of process where this transition fires. In our method, a business process is established based on the composition of web services. Therefore, it is reasonable to require a system to be sound. This paper adopts the notion of *weak soundness* that is slightly different from the soundness in [33].

Definition 4 (Weak Soundness): A composable workflow net CSWN = ISWN$_1 \oplus$*ISWN*$_2$ *is weakly sound if:*

1) *For each reachable marking (starting at* m_0*) the final marking* m_e *is reachable; and*
2) *For each reachable marking m such that* $m \geq m_e$ *holds* $m = m_e$.

Note that the requirements in this definition are a subset of the requirements in the soundness definition. Therefore, every sound service workflow net is weakly sound. Due to the interaction protocol incompatibility, two or more *ISWN*s with complementary functions cannot interact with each other. Then, a controller can be proposed as a middlebox and eliminate the incompatibility between them. Then, based on the former definition of *CSWN*, this sections presents the controller-based composition of service workflows net (*C-CSWN*).

Definition 5 (Controller-Based Compatibility): Given two ISWN$_1$*, ISWN*$_2$ *and a controller* $\mathbb{C}$*, if an existing CSWN has protocol incompatibility, the corresponding controller should be appended on exceptional message places in CSWN. That is, the composition of ISWN*$_1$ *and ISWN*$_2$ *is compatible with controller* $\mathbb{C}$ *if ISWN*$_1 \oplus \mathbb{C} \oplus$*ISWN*$_2$ *is weakly sound.*

Definitions 3–5 concern only the composition of two *ISWN*s. Our method can easily be extended to a multi-service composition scenario by stepwise composition and analysis.

Then, we first model service choreography for the prior scenario. We assume that the interfaces of services are completely matched. Later we show that this is not sufficient to ensure a successful service choreography. The *CSWN* is shown in Fig. 4, i.e., Messages *Product*, *Approve*, *Cancel*, *Payment*, *Visa* and *Master* are modeled with $p_{16}-p_{21}$, respectively. $P_{I1} = \{p_{1-5}\}$, $P_{I2} = \{p_{6-10}\}$ and $P_{I3} = \{p_{11-15}\}$ describe those of *ISWN*$_1$(**C**), *ISWN*$_2$(**S**) and *ISWN*$_3$(**T**), respectively. The operations are modeled by transitions, namely $T = \{t_{1-19}\}$ in *CSWN*. For example, Product Booking service initiates the communication by sending a *Product* and waits for payment, i.e., either *Visa* or *Master*, from client. By receiving the *Payment* from the Third Party Checkout service, the client solves an internal conflict and fixes the kind of payment. It can be easily proven: The composed service *ISWN*$_1 \oplus$*ISWN*$_2 \oplus$*ISWN*$_3$ is weakly sound. Thus, we call it is compatible.

Fig 5(a) shows two service workflow nets *ISWN*$_1$ and *ISWN*$_3$. The Third Party Checkout service solves an internal conflict and sends *Payment*. Thereafter, *ISWN*$_3$ is either in state p_{13} waiting for *Visa* or in state p_{14} waiting for *Master*. The client receives *Payment* and has the choice between the two kinds of payment. But, he does not know the internal state of the Third Party Checkout service. Thus, he might choose the "wrong" alternative. The composed service *ISWN*$_1 \oplus$*ISWN*$_3$ may end in a deadlock, thus it is not weakly sound. The two service workflow nets are not compatible, and a non-local choice mismatch issue arises.

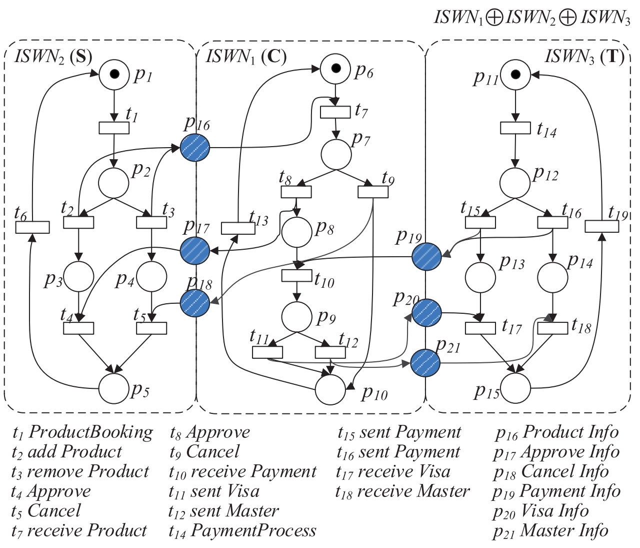


FIGURE 4. Illustration of an incompatible CSWN.

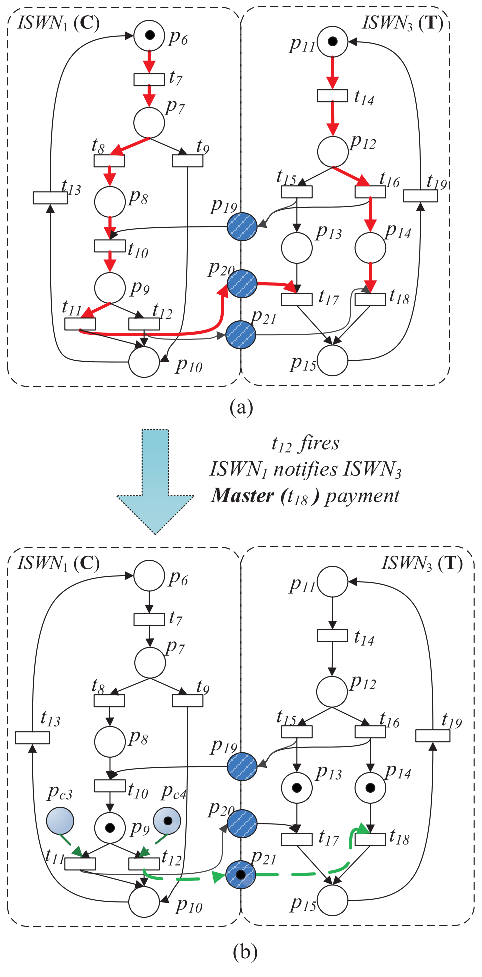


FIGURE 5. Controller-based service orchestration.

For example, if $ISWN_3$ decides to proceed *Master* (t_{18}) operation but $ISWN_1$ decides to proceed *Visa* (t_{11}) operation, then there is a deadlock after t_7, t_8, t_{10}, t_{11} and t_{14}, t_{16}, t_{18} fire (paths of fire are represented with bold arrows), as shown in Fig. 5(a). Then, in order to avoid a non-local choice mismatch, we append controlled places p_{c3} and p_{c4} to *Visa* (t_{11}) and *Master* (t_{12}) of service workflows $ISWN_1$ respectively when Third Party Checkout service is ready. If t_{18} fires in $ISWN_3$, we assure corresponding controls $u(p_{c3})$=0 and $u(p_{c4})$=1, respectively. So *MasterInfo* (p_{21}) is received by *Master* (t_{18}) operation of $ISWN_3$ after *Master* (t_{12}) fires in $ISWN_1$, as shown in Fig. 5(b). Therefore, by appending controller $\mathbb{C}$ to *CSWN*, we obtain a compatible service choreography. That is, *C-CSWN* is weakly sound.

IV. COMPATIBILITY ANALYSIS AND CONTROL POLICY

In this section we first present the compatibility analysis of service choreography. Then, a solution based on an optimal control policy is proposed to prevent protocol incompatibility in *CSWN*.

A. COMPATIBILITY ANALYSIS

In order to check the compatibility in service choreography, a set of states are now marked to denote the initial status of *CSWN* in Fig. 4. A naive method requires the construction of a reachability graph. However, the conventional reachability graph approach is inefficient or intractable, even for a bounded Petri net, due to state explosion in many practical applications. To alleviate the state explosion problem in the reachability graph of *CSWN*, the related definitions about reduced state space [34], [36] are presented. In our experiments, we have followed Valmaris heuristic [36] of selecting a stubborn set with a minimal number of transitions to build a reduced state space of the system.

Reduced state space is divided into deadlock-free states and deadlocks, as shown in Fig. 6. From Fig. 6, the set of forbidden (deadlock) states can be obtained, that is, M_F=$\{m_6, m_7, m_{15}, m_{18}\}$ which are represented by grey circles; the admissible (deadlock-free) states A_F=$\{m_{1-5}, m_{8-14}, m_{16}, m_{17}, m_{19}, m_{20}\}$ are represented by hollow circles; the set of deadlock transition domain (*DTD*) *DTD*=$\{t_8, t_9, t_{11}, t_{12}\}$ is shown via dashed directed arcs; the set of deadlock-free transition domain (*DFTD*) *DFTD*=$\{t_{2-7}, t_{10}, t_{13}, t_{15-19}\}$ is denoted by solid directed arcs.

Our control policy is to prevent any forbidden states from entering *DTD*, while ensuring that all other states within *DFTD* can still be reached. To do this we add some new places with an initial marking into *CSWN* such that the bad transitions can no longer occur from the critical good states of *DFTD*.

B. OPTIMAL CONTROL POLICY

State feedback policies for *CtlPN*s have been investigated by a number of researchers [37]–[39]. In this section we first consider state specifications, i.e., specifications given as a set

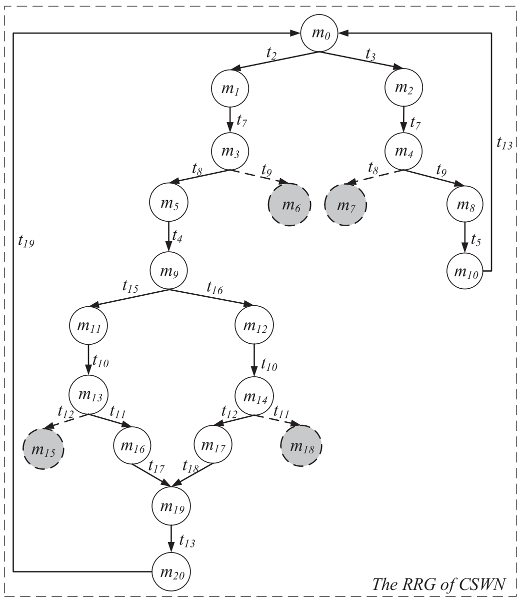


FIGURE 6. The reduced reachability graph (***RRG***) of the ***CSWN***.

of legal markings for *CSWN* to be controlled. In this setting the aim of the controller is that of restricting the behavior of a *CSWN* such that only legal markings can be reached. The corresponding control policy is called *state feedback control policy*. We show here how it can be computed for the *C-CSWN* model presented in Section III-B.

Given a *C-CSWN* with initial marking m_0, let M_F denote the set of *forbidden markings* (*states*). The main objective is to find a state feedback control policy U_F for which:

1) $R_\infty(U_F, m_0) \cap M_F = \emptyset$; and
2) for any policy U' such that $U' \geq U_F$, if U' satisfies 1) above, then $U' = U_F$.

We call a state feedback policy satisfying conditions 1) and 2) a *maximally permissive state feedback control policy* for the given forbidden state specification M_F.

A necessary and sufficient condition for the existence of a maximally permissive state feedback control policy is determined by an analysis of the *C-CSWN* behavior under the control u_{zero}. Specifically, define the set of *admissible markings* [27] for a *C-CSWN* with respect to a set of forbidden markings M_F as

$$A_F = \{m \in M \mid R_\infty(u_{zero}, m) \cap M_F = \emptyset\} \tag{4}$$

An existing state feedback control policy can ensure that a *C-CSWN* is kept out of the given forbidden markings. Then, sufficient and necessary conditions are further presented [39] as follows.

Theorem 1 [39, Th. 2]: For a C-CSWN with a forbidden specification of marking M_F and initial marking m_0, a single state feedback policy that is maximally permissive exists if and only if $m_0 \in A_F$.

In *C-CSWN*, multiple transitions are allowed to simultaneously fire. Thus, the single state feedback policy that is maximally permissive in Theorem 1 is nondeterministic.

In the state feedback policy that is maximally permissive, any state transitions to markings that are not in A_F are not allowed when the initial marking meets the condition $m_0 \in A_F$ in Theorem 1. Besides, the control set is changed at every state transition, and therefore only immediately reachable markings should be considered in specifying the admissible control set from a given marking. This leads to the following Theorem 2 that is similar to an extension of the supervisory control proposed by Ramadge and Wonham [35] in the *C-CSWN*.

Theorem 2 [39, Th. 3]: for a C-CSWN with an initial condition $m_0 \in A_F$ and a forbidden specification of marking, if $m \in R_\infty(U_F, m_0)$, then

$$U_F(m) = \{u \in U | R_1(u, m) - A_F = \emptyset\} \tag{5}$$

Then, state feedback policies that are maximally permissive for state specifications that are forbidden have been characterized. Next, for a given marking, how to calculate the admissible control set becomes the key problem. In this work, the first control objective is to prevent any forbidden markings M_F from being reached. Given an initial marking $m_0 \in A_F$, we know that u_{zero} will prevent any forbidden conditions from being satisfied. However, u_{zero} is often more restrictive than necessary. Our second control objective is to find the set of maximally permissive state feedback controls $u \in U_F$ for each marking $m \in A_F$ to guarantee that the state of the system under it will remain within the set A_F.

C. CONTROLLER EXISTENCE CHECKING

In order to check whether there exists a controller to compose partially compatible services into a weakly sound service, we introduce the method of controlled reduced reachability graph (*CRRG*). Its basic idea is to construct the *CRRG* of multiple service choreography, using a controller mechanism. A controlled place, transition and edge are constructed into the mechanism among multiple services choreography. The construction method of a *CRRG* is given in Algorithms 1 and 2.

1) STRATEGY OF CRRG

Firstly, derive a reduction state reachability graph, *RRG* according to *CSWN*. Secondly, decide if there exists a deadlock state in *RRG*; and if so, provide control-enabled state for each deadlock. Thirdly, derive a maximally permissive state feedback control policy U_F. Finally, derive the *CRRG* of multiple services choreography.

For each *ISWN*, the set of markings contains the reachable one within k state transitions from marking m under control u (k must be large enough). If a whole *CSWN* is composed of n *ISWN*s where each *ISWN* includes at most k execution steps, then its execution state space can roughly be k^n given a limited number of execution steps in each *ISWN*. This work adopts the method of a stubborn set, where the unrelated part is reduced via usability examination. The state space can be reduced to nk. Thus, the complexity of Algorithms 1 and 2 becomes $O((|V|+|E|) \times nk)$.

Algorithm 1 Method to Construct *CRRG*-Part 1

```
Input:
  $ISWN_i = (P_i, T_i, F_i)$, $i = 1, 2$; the stubborn set $S$
Output: $CRRG(m, U_F) = (V^{\mathbb{C}}, E^{\mathbb{C}})$ where $V^{\mathbb{C}}$ is a set of system states in C-CSWN, and $E^{\mathbb{C}}$ is a set of controlled transitions
  1: Construct $RRG(CSWN) = (V, E)$ where $V$ is a set of system states, and $E$ is a set of transitions that change system states.
    1.1 Initialize $(V, E) = (\{m_0 = m_{01} \times m_{02}\}, \emptyset)$; $m_0$ is an untagged node;
    1.2 If there are no untagged nodes in $V$, go to 2, and else go to 1.3;
    1.3 While there are untagged nodes in $V$, do
      1.3.1 Select an untagged node $m \in V$, and tag it "true" ($m = m_1 \times m_2$);
      1.3.2 For each enabled transition $t \in S$ at $m$,
        a) Compute $m' : m \xrightarrow{t} m'$;
        b) If $\exists m' \in V$, $E = E \cup \{(m, t, m')\}$, go to 1.4;
        c) If $\exists m'' \in V$, such that $\forall p \in P$, $m'' \xrightarrow{\delta} m'$,$(m'' \leq m') \wedge (m'' \neq m')$ and $\exists p \in P \wedge p \notin P_M$, s.t. $m'' < m$ then an unbounded reachability graph is detected;
        d) Else if $\exists m'' \notin V$, such that $m'' = m'$, then $V = V \cup \{m'\}$, $E = E \cup \{(m, t, m')\}$, tag $m'$ is "false" ($m'$ is untagged);
    1.4 According to the three preconditions of a stubborn set in [36], construct $RRG$, and go to 1.2;
```

According to Algorithms 1 and 2, *CRRG* of a *C-CSWN* can be obtained. The following theorem shows that U_F as defined above is the maximally permissive state feedback controller for the set of forbidden markings M_F.

Theorem 3: Given a live C-CSWN with a specification of forbidden markings set M_F, U_F as defined in Algorithms 1 and 2 is a maximally permissive state feedback control policy for the forbidden states avoidance problem of the C-CSWN with the forbidden state specification.

Proof: Algorithms 1 and 2 eliminate no good markings at each iteration when a new controller is added based on the optimal control policy. Thus, all good markings are preserved, leading to the optimally controller-based *CSWN* that is live. ■

From Theorem 1, the following theorem can be obtained, if controller $\mathbb{C}$ exists, the *ISWN*s with protocol incompatibility may be composed. According to Algorithms 1 and 2, we give the method to derive the state space of $ISWN_1 \oplus \mathbb{C} \oplus ISWN_2$. We will prove that if $ISWN_1 \oplus \mathbb{C} \oplus ISWN_2$ is weakly sound, there exists an optimal controller $\mathbb{C}$. We have the following theorem.

Theorem 4: Given $ISWN_i$, $i \in \{1,2\}$, and maximally permissive state feedback control $u \in U_F$, there exists an optimal controller $\mathbb{C}$ with respect to U_F, such that CSWN via $\mathbb{C}$ is compatible, iff $ISWN_1 \oplus \mathbb{C} \oplus ISWN_2$ is weakly sound, i.e.,

Theorem 2 can be easily proven. We therefore omit the proof here, it is similar to that in [24]. It claims that the

Algorithm 2 Method to Construct *CRRG*-Part 2

```
2: If $RRG$ contains any deadlock state, continue the following steps, and else no controller $\mathbb{C}$ exists, such that $u(p_c) = u_{zero}$, i.e., namely, $RRG(CSWN) = (V,E) = (V^{\mathbb{C}},E^{\mathbb{C}})$, go to 3;
  2.1 According to $RRG$, and the set of forbidden markings, $M_F$, compute $T_{\mathbb{C}} = \{t_{cs}|s = 1, 2\}$, for $p_c$ is the controlled place of $t_c$, thus $P_{\mathbb{C}} = \{p_{cr}|r = 1, 2\}$;
  2.2 The length of a marking vector for any node $m \in V$ is increased from $|P_I| + |P_M|$ to $|P_I| + |P_M| + |P_C|$;
  2.3 Fill in $RRG(CSWN) = (V,E)$ with the subvector value corresponding to $|P_{\mathbb{C}}|$. $\forall m', m'' \in V$, $V^{\mathbb{C}} = V \cup \Gamma_{p_c \in P_{\mathbb{C}}}(m)$, where $\Gamma_{p_c \in P_{\mathbb{C}}}(m)$ is the mapped subvector in $P_{\mathbb{C}}$ from $m$;
  2.4 $\forall m', m''$, if $(m', m'') \in E$, and each enabled controlled transition $t_c \in T_{\mathbb{C}}$ for $m$, $m' \xrightarrow{t_c} m''$, then $E^{\mathbb{C}} = E \cup \{(\Gamma_{p_c \in P_{\mathbb{C}}})(m'), T_{\mathbb{C}}, (\Gamma_{p_c \in P_{\mathbb{C}}})(m'')\}$;
3: Derive a maximally permissive state feedback control policy $U_F$,
  3.1 Compute the set of all controlled transitions $T_{\mathbb{C}}$, which are marked with $m$ in $C$-$CSWN$.
    3.1.1 If $T_{\mathbb{C}} = \emptyset$, compute the set of transitions $T$;
    3.1.2 If $\forall t_c \in T_{\mathbb{C}} \subset T$, then there exists $m[t_c\rangle m_f$;
    3.1.3 If $m_f \in M_F$, then there exists $t_c \in T_{\mathbb{C}}$;
  3.2 If $T_{\mathbb{C}} = \emptyset$, $u(p_c) = 1$. Otherwise, the control $u \in U_F$ can be obtained a maximally permissive state feedback control policy according to a state feedback policy satisfying conditions 1 and 2;
4: Return $CRRG(m, U_F)$.
```

method to generate *CRRG* is equivalent to the one that generates a reduced state space of the service choreography by using stubborn sets. Since we are only interested in terminal states in a state-space, the reduced state space eliminates some intermediate states, and preserves all the relevant properties.

V. CONTROLLER GENERATION APPROACH

A. OPTIMAL CONTROLLER GENERATION

In Section IV-C, we use *CRRG* to check the existence of a controller among the multiple *ISWN*s. This section gives the method to build it if existing.

According to Theorem 2, it can be concluded that *CSWN* is compatible, i.e., controller $\mathbb{C}$ exists. To address the above protocol incompatibility issue, this work proposes an Algorithm 3 based on the modification by appending additional controllers to *CSWN* as follows:

Continue with the prior example. By using the proposed method, we conclude that *CSWN*, the interaction of $ISWN_1$, $ISWN_2$ and $ISWN_3$, is compatible via controller $\mathbb{C}$, as shown in Fig. 7. Two or more incompatible *ISWN*s can be composable by appending an additional controller, which avoids other complicated methods. For example, substitution service environment-based methods [18] explore an entire reachability state space of service interaction, which encounters

Algorithm 3 Generation of the Optimal Controller

```
Input:
  $ISWN_i = (P_i, T_i, F_i), i = 1, 2$, and the maximally permissive state feedback control policy $U_F$
Output: Optimal controller $\mathbb{C} = (P_{\mathbb{C}}, T_{\mathbb{C}}, B)$
  1: If the interaction among $ISWN_i$, $i = 1,2$, leads to a deadlock, namely there exists a forbidden state $m_f$, then set $P_{\mathbb{C}} = \{p_{cr}|r = 1, 2\}$, $T_{\mathbb{C}} = \{t_{cs}|s = 1, 2\}$, $B \leftarrow \emptyset$, tag and push $u(p_c)=0$ into the $ST$ ($ST$ is stack);
  2: If $stack \neq \emptyset$, go to 3, else go to 6;
  3: If there exists an adjacent node from $ST$(top), and untagged $u$, and go to 4. Otherwise, go to 5;
  4: Define $\Delta u = u - ST(top)$. If there is $ST$(top)$\xrightarrow{t_{cs}} u$, then go to 4.1;
    4.1 Define $\gamma = \Delta u$, $\forall p_{cr} \in P_{\mathbb{C}}$, if $\gamma > 0$
      4.1.1 Add controlled transitions $\{t_{c1}, t_{c2}\}$ to $\mathbb{C}$, namely $T_{\mathbb{C}} \leftarrow T \cup \{t_{c1}, t_{c2}\}$; and controlled places $\{p_{c1}, p_{c2}\}$ to $\mathbb{C}$, namely $P_{\mathbb{C}} \leftarrow P \cup \{p_{c1}, p_{c2}\}$; and
      4.1.2 $B \leftarrow B \cup \{(p_{c1}, t_{c1}), (p_{c2}, t_{c2})\}$
    4.2 Tag and push $u$ into $ST$, and go to 3;
  5: Pop $ST$, and then go to 2;
  6: According to Algorithms 1 and 2, construct optimal controller $\mathbb{C}$;
  7: Return $\mathbb{C} = (P_{\mathbb{C}}, T_{\mathbb{C}}, B)$
```

the state space explosion problem. This method can only be adaptive to numerous services with the same interfaces but different interaction protocols, as more optional services can lead to more services with different interaction protocols, which means more likely to achieve *CSWN* with completely matched interaction patterns. Along with the increase of the above kind of services, it is more difficult to find a completely fit service, as one faulty behavior of a single *ISWN* can make the whole *CSWN* with many other *ISWN*s incompatible. For example, in a *CSWN* composed of n *ISWN*s, there are m optional services with same interfaces but different interaction protocols for each *ISWN*. It is unimaginably complex since with this method has m^n time complexity. The proposed strategy by appending an additional controller to *CSWN* is convenient, as it can modify incompatible parts in *ISWN* rather than replace the whole *ISWN* to meet the composibility requirement. Therefore, the policy of appending an optimal controller can be realized in linear time complexity.

B. HYBRID OPTIMAL CONTROLLER-BASED PROTOCOL COMPATIBILITY

In this section, our optimal control policy is constructed for *CSWN*. The focuses are on how to analyze a *CSWN* model and to provide a new policy when *CSWN* is incompatible, namely, an optimal control policy to avoid deadlock. Our work aims to obtain live, and thus deadlock-free, *CtlPN* models of *CSWN*, and at the same time to ensure the optimal use of the system resources. A control policy is defined as the addition of new constraints to the system such that its initial

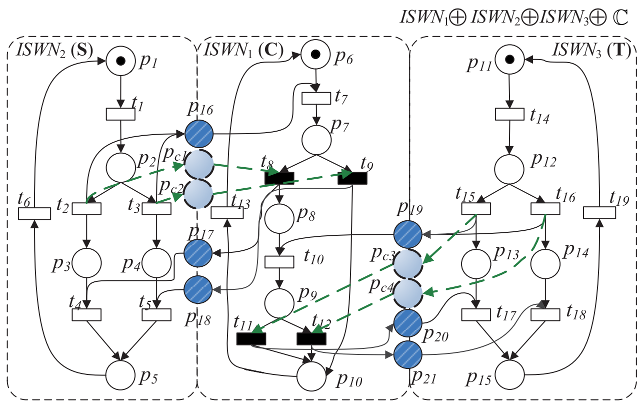


FIGURE 7. ***CSWN* based on Controller, which is live, i.e. deadlock-free, and optimal, i.e. maximally permissive.**

behavior is restricted to a set of states considered as good states, which allow the system to evolve without reaching a deadlock. At the same time, it should be ensured that all possible good states of the systems can still be reached.

The example of incompatible web services is as shown in Fig. 3. The collaboration business processes can be described in WS-CDL. It is deadlock-prone and the *ISWN*s are incompatible. According to the aforementioned method, the optimal controller $\mathbb{C}$ is constructed among multiple interaction services, as shown in Fig. 7 (controlled transition use black rectangle). In Section 4.3, the weak sound *C-CSWN* is constructed. That is, optimal controller-based service choreography is compatible.

According to Fig. 6, four deadlocks states are obtained in blue circles, namely $M_6(p_3,[0,0,1],p_{10},[0,0,0],p_{12})$, $M_7(p_4,[0,1,0],p_8,[0,0,0],p_{12})$, $M_{18}(p_2,[0,0,0],p_{10},[0,0,1],p_{13})$, and $M_{19}(p_2,[0,0,0],p_{10},[0,1,0],p_{14})$ within *DTD*, that is, the four deadlocks states of *RRG*, namely $\{p_3, p_{18}, p_{10}, p_{12}\}$, $\{p_4, p_{17}, p_8, p_{12}\}$, $\{p_2, p_{10}, p_{21}, p_{13}\}$, and $\{p_2, p_{10}, p_{20}, p_{14}\}$. The other states are within *DFTD*. According to the proposed method, controller $\mathbb{C}$ among *ISWN*₁-*ISWN*₃ is constructed, as shown in Fig. 7. Corresponding controlled places are $P_{\mathbb{C}} = \{p_{c1-c4}\}$, which are denoted by light blue circles; controlled transitions are $T_{\mathbb{C}} = \{t_8, t_9, t_{11}, t_{12}\}$, which are denoted by black rectangles. Controlled places and transitions can be connected by controlled arcs with dashed arcs, namely $B = \{(t_2, p_{c1}), (p_{c1}, t_8), (t_3, p_{c2}), (p_{c2}, t_9), (t_{15}, p_{c3}), (p_{c3}, t_{11}), (t_{16}, p_{c4}), (p_{c4}, t_{12})\}$. Then, it is obtained that the set of maximally permissive state feedback control policies for forbidden markings $M_F = \{m_6, m_7, m_{18}, m_{19}\}$ respectively is:

$$U_F(m) = \{u_f, f = 1, 2, 3, 4, 5\}$$

where

$$\begin{aligned}
u_1 &: u_1(p_{c1}) = 1, \quad u_1(p_{c2}) = 0, \; u_1(p_{c3}) = 0, \; u_1(p_{c4}) = 0\\
u_2 &: u_2(p_{c1}) = 1, \quad u_2(p_{c2}) = 0, \; u_2(p_{c3}) = 1, \; u_2(p_{c4}) = 0\\
u_3 &: u_3(p_{c1}) = 1, \quad u_3(p_{c2}) = 0, \; u_3(p_{c3}) = 0, \; u_3(p_{c4}) = 1\\
u_4 &: u_4(p_{c1}) = 0, \quad u_4(p_{c2}) = 1, \; u_4(p_{c3}) = 0, \; u_4(p_{c4}) = 0\\
u_5 &: u_5(p_{c1}) = 0, \quad u_5(p_{c2}) = 0, \; u_5(p_{c3}) = 0, \; u_5(p_{c4}) = 0
\end{aligned}$$

Through the set of maximally permissive state feedback control policies U_F, the controlled places, transitions and arcs, and an optimal controller $\mathbb{C}$ are constructed where $\mathbb{C} = \{p_{c1-c4}, T_{\mathbb{C}}, B\}$. Therefore, the compatible *CSWN* based on the controller is obtained.

Finally, we transform four additional controlled channels to collaboration business processes. For example, controller p_{c1} means that *ISWN*₂ is informed to *Approve* booking operation by *ISWN*₁. Controller p_{c2} means that *ISWN*₂ is informed to *Cancel* booking operation by *ISWN*₁. Controller p_{c3} means that *ISWN*₃ is informed to use *Visa* payment by *ISWN*1₁. Controller p_{c4} means that *ISWN*₃ is informed to use *Master* payment by *ISWN*₁. According to Theorem 2, the new collaboration business processes are compatible.

VI. RELATED WORK

In order to analyze the compatibility of service choreography, several formal semantics have been adopted, such as graphs [40], message sequence charts (MSC) [41], ontology [3], finite state machines (FSM) [43], process algebra (PA) [47], martin type theory (MTT) [49] and Petri nets (PN) [50]–[52]. In particular, Petri net models have been widely adopted to describe service choreography, from languages like WS-CDL to interaction Petri nets [25]. Therefore, it is appropriate for modeling and analyzing compatibility and equivalence of web services. In our work, we use the analysis methods of *CtlPN*. This section gives an overview of the related works in this research area and makes comparisons among them.

A. COMPATIBILITY VERIFICATION

The existing work can be divided into two categories, i.e., syntactic and protocol compatibility verification. For the syntactic consideration, Nezhad *et al.* [15] identify the split/merge interface mismatch and construct an adaptor for these services. Zhou *et al.* [40] gain abstract protocols from service protocols by defining many rules. Then, to adapt two services, they establish an adaptation matrix through a depth-first search with back tracking methods. Foster *et al.* [41] propose a model-based approach to verify web service composition by using MSC. The work only deals with compatibility in the message interface level. To establish a mapping between the interfaces from the automated synthesis of mediators, Bennaceur and Issarny [42] present a method to combine constraint programming and ontology reasoning. Based on FSM, Bachir and Fauvet [43] check whether two services are incompatible syntactically, and locate interfaces of services where incompatibilities happen.

Even if service interfaces match perfectly, interaction protocol mismatch may occur. Most researchers work on choreography modeling using Petri nets and a stitched interaction protocols approach [44], [45]. Interaction protocols can be represented using workflow nets, which are a subclass of Petri nets where a distinction is made between internal, input and output places [46]. Input and output places represent inbound and outbound message queues for communication with the environment. Different workflow nets are stitched together for carrying out compatibility checking. π-calculus is a popular process algebra especially suited for describing interacting processes [47]. In [48], the interaction modeling based on process algebra (PA) has been driven by the WS-CDL. Martens [18] transforms the correctness check of the composed processes into the usability and compatibility check, and presents algorithms to verify these properties locally. Their approach yields a concrete example how to use a given Web services. In [49], Yin *et al.* model Web service behaviors with martin type theory (MTT), and then propose a consistency and compatible judgment method of Web service behaviors.

The above methods can address the issue of compatibility verification for a small number of services. However, existing methods only identify a deadlock if any, but do not offer anything to resolve it. We go a step further in terms of avoiding the occurrence of deadlocks, by appending an optimal controller to any deadlock prone states for behaviours incompatibility. We also adopt a reduced state space method to overcome the state space explosion problem when verifying the composition of complex services.

B. DEADLOCK PREVENTION OF PETRI NETS

Deadlock avoidance can be achieved by characterizing the compatibility in terms of structural Petri net objects, e.g., siphons. Our previous work [19] presents a technique to hold tokens in siphons according to additional information channels, such that each siphon is always marked. In this case, by adding new net elements to the initial Petri net of service choreography, the control policy can be implemented. The addition of new net elements causes that in order not to generate a deadlock state, some enabled transitions are stopped. Though the online computation of structural analysis technology of Petri net is quick, to achieve its deadlock-free state, it may eliminate several good states of a system. Thus, the controlled model of the system may not be maximally permissive.

By checking the reachability graph, it is also possible to obtain a deadlock-free system. For example, Li and Wonham [50] establish relationships between the language of the *CtlPN* (sequences of transitions) and predicates on the state space for a *CtlPN* (i.e., sets of allowable markings) in state feedback control. They show balanced controllers that are maximally permissive from the given initial marking to the set of reachable markings. In [51], a Petri net-based state reachability graph is presented to prevent deadlocks in the application of web services to distributed business processes. Besides, though the problem of compatibility in interaction of web services is similar to the traditional deadlock problem in flexible manufacturing systems (FMS) [52], it is not totally the same. Therefore, these approaches cannot well solve the protocol incompatibility problem in service choreography. In addition, these approaches may face the problem of state explosion for a sizable Petri net.

TABLE 1. Related works on web service composition compatibility.

Related work	FM	CM	SE	MP	CP	CTP	OM
Nezhad [15]	FSM	+	-	+	-	+	-
Martens [18]	PN	+	+	+	-	-	-
Xiong [19]	PN	+	-	-	+	+	-
Foster [41]	MSC	-	N/A	+	-	-	-
Our approach	CtlPN	+	-	-	+	+	+

Compared with these researches, the structural analysis is combined with the state analysis of reachability in this work. An optimally controlled *C-CSWN* model is established for service choreography, and this can guarantee the high performance of the resulting service choreography. Thus, the approach in this paper is both efficient and effective in this sense.

A summary of service composition compatibility is shown in Table 1. The columns of the table correspond to the following criteria, where (-) means not at all and (+) means fully.

- **FM** denotes the formalization methods used: CtlPN for Controlled Petri Nets and PN for plain (i.e. low-level) Petri Nets, MSC for Message Sequence Charts, FSM for Finite State Machines.
- **CM** denotes whether the formalization considers the composition of several business processes.
- **SE** denotes whether there is the state space explosion problem in the service composition.
- **MP** denotes whether the modification policies of incompatibility conditions are given.
- **CP** denotes whether the control policies of incompatibility conditions are proposed.
- **CTP** denotes whether the correctness proof of a method is proposed.
- **OM** denotes whether the optimal approach is proposed in the incompatibility of service choreography.

VII. CONCLUSIONS

When multiple web services provide complementary functionality and can be linked together in principle, but their interaction patterns do not fit each other exactly, they cannot be directly composed in the Internet-scale service ecosystems. The challenge is to analyze the compatibility of services and automatically compose them with the minimum engineering cost.

To tackle this challenge we first model a service choreography as a controlled Petri net. Based on this model, the incompatibility of multiple services is analyzed with a reduced reachability analysis technique. Afterward a controlled reduced reachability graph is constructed by combining structural and reachability analyses, obtaining deadlock and deadlock-free states, and developing a maximally permissive state feedback control policy to resolve deadlocks. As a result the required compatibility in service choreography is enforced.

Our future work aims at designing an effective controller under distributed service choreography. We also plan to apply other methods of avoiding the state explosion problem, e.g., partial orders and unfoldings methods, to simplify the state space while preserving the concerned properties such as boundedness and liveness.

REFERENCES

[1] H. Mei, G. Huang, and T. Xie, "Internetware: A software paradigm for internet computing," *Computer*, vol. 45, no. 6, pp. 26–31, Jun. 2012.

[2] A. Bertolino, M. B. Blake, P. Mehra, H. Mei, and T. Xie, "Software engineering for internet computing: Internetware and beyond [guest editors' introduction]," *IEEE Softw.*, vol. 32, no. 1, pp. 35–37, Jan./Feb. 2015.

[3] A. Bennaceur and V. Issarny, "Automated synthesis of mediators to support component interoperability," *IEEE Trans. Softw. Eng.*, vol. 41, no. 3, pp. 221–240, Mar. 2015.

[4] W. Tan, J. Zhang, and I. Foster, "Network analysis of scientific workflows: A gateway to reuse," *IEEE Comput.*, vol. 43, no. 9, pp. 54–61, Sep. 2010.

[5] OASIS. *Web Services Business Process Execution Language Version 2.0.* [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html

[6] M. Trainotti *et al.*, "Astro: Supporting composition and execution of Web services," in *Proc. Int. Conf. Service-Oriented Comput. (ICSOC)*, vol. 3826. 2005, pp. 495–501.

[7] W3C. *Web Services Choreography Description Language Version 1.0.* [Online]. Available: http://www.w3.org/TR/ 2004/WD-wscdl-10-20041217

[8] W3C. *Web Service Choreography Interface (WSCI) 1.0.* [Online]. Available: http://www.w3.org/TR/wsci/

[9] M. Autili, P. Inverardi, and M. Tivoli, "Automated synthesis of service choreographies," *IEEE Softw.*, vol. 32, no. 1, pp. 50–57, Jan. 2015.

[10] M. B. Blake, W. Tan, and F. Rosenberg, "Composition as a service," *IEEE Internet Comput.*, vol. 14, no. 1, pp. 78–82, Jan./Feb. 2010.

[11] S. Basu, T. Bultan, and M. Ouederni, "Deciding choreography realizability," in *Proc. 39th Annu. ACM SIGPLAN-SIGACT Symp. Principles Program. Lang.*, Jan. 2012, pp. 191–202.

[12] M. B. Blake, "Decomposing composition: Service-oriented software engineers," *IEEE Softw.*, vol. 24, no. 6, pp. 68–77, Nov. 2007.

[13] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch: Why reuse is still so hard," *IEEE Softw.*, vol. 26, no. 4, pp. 66–69, Jul. 2009.

[14] W. Kongdenfha, H. R. Motahari-Nezhad, B. Benatallah, F. Casati, and R. Saint-Paul, "Mismatch patterns and adaptation aspects: A foundation for rapid development of Web service adapters," *IEEE Trans. Services Comput.*, vol. 2, no. 2, pp. 94–107, Apr. 2009.

[15] H. R. M. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, "Semi-automated adaptation of service interactions," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 993–1002.

[16] D. M. Yellin and R. E. Strom, "Protocol specifications and component adaptors," *J. ACM Trans. Prog. Lang. Syst.*, vol. 19, no. 2, pp. 292–333, 1997.

[17] H. Ben-Abdallah and S. Leue, "Syntactic detection of process divergence and non-local choice in message sequence charts," in *Proc. 3rd Int. Workshop Tools Algorithms Construct. Anal. Syst.*, 1997, pp. 259–274.

[18] A. Martens, "Usability of Web services," in *Proc. 4th Int. Conf. Web Inf. Syst. Eng. Workshops*, 2003, pp. 182–190.

[19] P. C. Xiong, Y. S. Fan, and M. C. Zhou, "A Petri net approach to analysis and composition of Web services," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 2, pp. 376–387, Mar. 2010.

[20] P. C. Xiong, C. Pu, and M. C. Zhou, "Protocol-level service composition mismatches: A Petri net siphon based solution," *Int. J. Web Services Res.*, vol. 7, no. 4, pp. 1–20, 2010.

[21] W. V. D. van der Aalst and K. V. van Hee, *Workflow Management: Models, Methods, and Systems*. Cambridge, MA, USA: MIT Press, 2002.

[22] M. P. Fanti and M. Zhou, "Deadlock control methods in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 1, pp. 5–22, Jan. 2004.

[23] P. Xiong, Y. Fan, and M. Zhou, "QoS-Aware Web service configuration," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 888–895, Jul. 2008.

[24] W. Tan, Y. S. Fan, and M. C. Zhou, "A Petri net-based method for compatibility analysis and composition of Web services in business process execution language," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 94–106, Jan. 2009.

[25] G. Decker and M. Weske, "Local enforceability in interaction Petri nets," in *Proc. 5th Int. Conf. Brisbane, Bus. Process Manage. (BPM)*, Berlin, Germany, 2007, pp. 305–319.

[26] E. G. Coffman, M. J. Elphick, and A. Shoshani, "System deadlocks," *Comput. Surv.*, vol. 3, no. 2, pp. 67–78, 1971.

[27] B. H. Krogh, "Controlled Petri nets and maximally permissive feedback logic," in *Proc. 25th Allerton Conf. Commun. Control Comput.*, 1987, pp. 317–326.

[28] A. Ichikawa and K. Hiraishi, "Analysis and control of discrete event systems represented by Petri nets," in *Proc. Discrete Event Syst. Models Appl. IIASA Conf. Sopron, Hungary*, Berlin, Germany, 1988, pp. 115–134.

[29] G. Stremersch and R. K. Boel, "Reduction of the supervisory control problem for Petri nets," *IEEE Trans. Autom. Control*, vol. 45, no. 12, pp. 2358–2363, Dec. 2000.

[30] A. Ghaffari, N. Rezg, and X. L. Xie, "Design of a live and maximally permissive Petri net controller using the theory of regions," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 137–141, Feb. 2003.

[31] W. T. Tsai, P. Zhong, X. Bai, and J. Elston, "Dependence-guided service composition for user-centric SOA," *IEEE Syst. J.*, vol. 8, no. 3, pp. 889–899, Sep. 2014.

[32] J. Zhang, D. Kuc, and S. Lu, "Confucius: A tool supporting collaborative scientific workflow composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 2–17, Jan./Mar. 2014.

[33] W. M. P. van der Aalst, "Structural characterizations of sound workflow nets," *Comput. Sci. Rep.*, vol. 96, no. 23, pp. 18–22, 1996.

[34] P. Godefroid and P. Wolper, "Using partial orders for the efficient verification of deadlock freedom and safety properties," in *Proc. 3rd Int. Workshop Comput. Aided Verification*, Jul. 1991, pp. 332–342.

[35] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[36] A. Valmari, "Stubborn sets for reduced state space generation," *Adv. Petri Nets*, vol. 483, pp. 491–515, 1990.

[37] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Trans. Autom. Control*, vol. 35, no. 5, pp. 514–523, May 1990.

[38] Y. Li and W. M. Wonham, "Control of vector discrete-event systems. II. Controller synthesis," *IEEE Trans. Autom. Control*, vol. 39, no. 3, pp. 512–531, Mar. 1994.

[39] L. E. Holloway, B. H. Krogh, and A. Giua, "A survey of Petri net methods for controlled discrete event systems," *Discrete Event Dyn. Syst.*, vol. 7, no. 2, pp. 151–190, 1997.

[40] Z. Zhou, S. Bhiri, H. Zhuge, and M. Hauswirth, "Assessing service protocol adaptability based on protocol reduction and graph search," *Concurrency Comput. Pract. Exper.*, vol. 23, no. 9, pp. 880–904, 2011.

[41] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Tool support for model-based engineering of Web service compositions," in *Proc. IEEE Int. Conf. Web Services*, Jul. 2005, pp. 95–102.

[42] A. Bennaceur and V. Issarny, "Automated synthesis of mediators to support component interoperability," *IEEE Trans. Softw. Eng.*, vol. 41, no. 3, pp. 221–240, Mar. 2015.

[43] A. A. Bachir and M. C. Fauvet, "Diagnosing and measuring incompatibilities between pairs of services," in *Proc. 20th Int. Conf. DEXA*, 2009, pp. 229–243.

[44] R. Dijkman and M. Dumas, "Service-oriented design: A multi-viewpoint approach," *Int. J. Cooperat. Inf. Syst.*, vol. 13, no. 4, pp. 337–368, 2004.

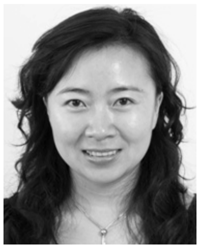
[45] W. M. P. van der Aalst and M. Weske, "The P2P approach to interorganizational workflows," in *Proc. 13th Conf. Adv. Inf. Syst. Eng.*, 2001, pp. 140–156.

[46] A. Martens, "Analyzing Web service based business processes," in *Proc. 8th Int. Conf. Fundam. Approaches Softw. Eng.*, 2005, pp. 19–33.

[47] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes," *Inf. Comput.*, vol. 100, no. 1, pp. 1–40, 1992.

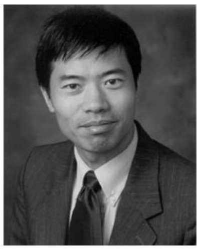
[48] M. Carbone, K. Honda, and N. Yoshida, "Structured communication-centred programming for Web services," in *Proc. 16th Eur. Symp. Program.*, 2007, pp. 2–17.

[49] Y. Y. Yin, Y. Li, S. G. Deng, and J. W. Yin, "Determining on consistency and compatibility of Web service behavior," *Acta Electron. Sin.*, vol. 37, no. 3, pp. 433–438, 2009.
[50] Y. Li and W. M. Wonham, "Control of vector discrete-event systems. I. The base model," *IEEE Trans. Autom. Control*, vol. 38, no. 8, pp. 1214–1227, Aug. 1993.
[51] S. Narayanan and S. A. Mcllraith, "Simulation, verification and automated composition of Web services," in *Proc. 11th Int. Conf. World Wide Web*, 2002, pp. 77–88.
[52] Y. Chen, Z. Li, and M. Zhou, "Optimal supervisory control of flexible manufacturing systems by Petri nets: A set classification approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 549–563, Apr. 2014.



JING BI (M'13–SM'16) received the Ph.D. degree from Northeastern University, Shenyang, China, in 2011. From 2009 to 2010, she was involved in cloud computing with the IBM China Research Laboratory. From 2013 to 2015, she was a Post-Doctral Researcher with the Department of Automation, Tsinghua University, China. She is currently an Associate Professor with the School of Software Engineering, Beijing University of Technology, China. Her research interests include service computing, cloud computing, large-scale data analytics, and resource optimization. He was a recipient of the IBM Ph.D. Fellowship Award in 2009.

HAITAO YUAN (S'15–M'16) received the B.S. and M.S. degrees in software engineering from Northeastern University, Shenyang, China, in 2010 and 2012, respectively, and the Ph.D. degree in control science and engineering from Beihang University, Beijing, China, in 2016. He was a Visiting Scholar with the New Jersey Institute of Technology, Newark, NJ, USA, in 2015. He is currently an Assistant Professor with the School of Software Engineering, Beijing Jiaotong University, Beijing, China. His research interests include cloud computing, resource allocation, software-defined networking, and energy efficiency. He was a recipient of the Google Excellence Scholarship in 2011.



MENGCHU ZHOU (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990. He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990. He is currently a Distinguished Professor of Electrical and Computer Engineering. He has authored over 680 publications, including 12 books, over 360 journal papers (over 260 in IEEE Transactions), and 28 book chapters. His research interests are in Petri nets, Internet of Things, big data, web services, manufacturing, transportation, and energy systems He is the Founding Editor of the IEEE *Press Book Series on Systems Science and Engineering*. He is a recipient of the Humboldt Research Award for U.S. Senior Scientists, the Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from the IEEE Systems, Man, and Cybernetics Society. He is a Life Member of Chinese Association for Science and Technology, USA, where he served as the President in 1999. He is a fellow of the International Federation of Automatic Control and the American Association for the Advancement of Science.