TELKOMNIKA, Vol.14, No.2, June 2016, pp. 734~740
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v14i1.2732

Property Analysis of Composable Web Services

Wei Liu*, Lu Wang, Yuyue Du, Pin Wang, Chun Yan
State Key Laboratory of Mining Disaster Prevention and Control,
Shandong University of Science and Technology, Qingdao, China
College of Information Science and Engineering, Shandong University of Science & Technology, Qingdao, China
The Key Laboratory of Embedded System and Service Computing, Ministry of Education,
Tongji University, Shanghai, China
*Corresponding auther, e-mail: liuwei_doctor@yeah.net

Abstract

Web services are basic components constructing a flexible business process software. By composing multiple Web services, a complicated business process across organization and departments can be formed. This paper present a formal model of composable Web services: composed service process nets (CSPNs). Properties of Composable Web services are analyzed based on CSPNs. The relation between a CSPN and its corresponding Web service process subnets are discussed. The property analysis methods of CSPNs are come up with. A dynamic property of CSPNs can be determined based on static net structures by means of the proposed methods.

Keywords*: passing messages, deadlock-freedom, business process, Web services*

1. Introduction

Service-oriented computing (SOC) is widely used [1]. With the development of Web services, the number of Web services is increasing dramatically [2-3]. Web services become a basic component constructing a flexible business process. By composing multiple Web services, a complicated business process can be formed [4-5]. The application of SOC reduces the complexity of business integration within an enterprise or organization and across enterprises or organizations [6]. Service-oriented computing (SOC) changed the tranditional software development. It has been used in e-commerce of across organizations and within an organization.

Different enterprises or organizations collaborate mutually and form inter-enterprise or inter-organization workflow. When collaboration happens among enterprises or organizations and different departments within an enterprise or organization, a workflow across organizations or departments is generated by means of interactions [7-8]. Each organization or department is served by a Web service and this Web service serve as a business process among organizations or departments within an organization [9]. Web services also can be combined by many small Web services within an enterprise or a workflow among departments of an enterprise formed by interactions of different departments [10]. In SOC architecture, the interactions or collaborations among different enterprises or departments are implemented by message passing among Web services. By means of message passing, Web services in different enterprises or departments are connected forming inter-organizational Web services interaction system, which leads to Web service composition across organizations. Web service composition is a important and valuable reserach field in SOC [11].

To realize Web service composition effectively, many composition languages and interface specifications are proposed, such as WS-BPEL, WS-CDL and WSCI [12]. They can be used to describe Web service composition. But they lack formal analysis for Web service composition.

Whether Web services are composable or not depends on the proper exection of Web service composition workflow across orgnizations. If Web service composition workflow can be executed properly, Web services are composable across orgnizations. If Web service composition workflow cannot be executed properly, Web services are not composable across

Received September 29, 2015; Revised January 10, 2016; Accepted Feb 3, 2016

orgnizations. Hence, to ensure Web services are composable, the proper exection of Web service composition workflow need be analysized and verified. The research in this area is paid more attention in SOC. We focus on and study this topic in this paper.

Web service composition workflow is often analyzed by formal methods including Petri nets [13], state automaton [14], description logic [15], process algebra [16] and so on. Process algebra is unintuitive for its excessive symbolic expression. For description logic, modeling a service composition is still at a theoretical level since it is complex for describing services and analyzing their composition. State automaton is intuitive, but has less mathematical analysis methods. Based on their strict mathematical definitions and graphical expression, Petri nets are an effective description and analysis tool for modeling services and fit for modeling distributed and loose-coupled systems like Web services. Hence, Petr nets are used to model Web service workflow and Web service composition workflow in this paper.

In this paper, the internal process of a Web service is represented by a service process net (SPN) based on Petri nets. The interactions among multiple Web services are denoted by a composed service process net (CSPN). A CSPN is composed by common passing message places in different SPNs. The interface passing message places in a SPN indicate the passing messages among Web services. The collaborations and interactions among Web services are represented by CSPNs.

By CSPNs and SPNs, We present necessary conditions of proper execution of Web service composition workflow. By these necessary conditions, improper execution of Web service composition can be judged. Even though each of multiple Web services is deadlock-free, a composition workflow of multiple Web services is not certain to deadlock-free. We propose the necesssary condition of the proper execution of a composition workflow of multiple deadlock-free. We give the subclass of Web service compsition. For the subclass, we give the sufficent and necessary condition of proper execution of composition workflow. Different from previous methods, these methods presented in this paper are based on the static structure of CSPNs. By these methods, it is esay to analyze properties of composable Web services.

The rest of this paper is organized as follows. In Section 2, Formal models of composable Web services is presented and introduced. We analyze properties of composable Web services in detail in Section 3. Section 4 concludes the paper.

2. Formal Models of Composable Web Services

Formal models are proposed for studying the satisfied properties of composable Web services. The Web service process nets are put forwarded to denote Web services. The internal service process nets (ISPNs) are raised to signify the inner Web service process. And The composed Web service process nets (CSPNs) are presented to express composed Web services.

Definition 1: A Web service is executed properly iff its input passing messages can be gotten and it can terminate normally.

Definition 2: A composed Web service is executed properly iff each Web service in composed service is executed properly.

A Web service process can be formally represented by a SPN model.

Definition 3: $SPN=(P, T, F, M_0)$ is a Web service process net where

(1) P is a set of places. $P=P_I \cup P_{MI} \cup P_{MO}$ are disjunct subsets of places, where P_I is a set of internal places, P_{MI} is a set of input passing message places and P_{MO} is a set of output passing message places;

(2) P includes two special internal places: i and o. Place i is a start place: ${}^\bullet i=\phi$. Place o is a end place: $o^\bullet=\phi$;

(3) T is a set of transitions and $P \cap T=\phi$. $\forall t \in T$, ${}^\bullet t$ and $t^\bullet$ separately includes a internal place at least;

(4) $F \subseteq (P \times T) \cup (T \times P)$ are directed edges representing flow relations;

(5) M: $P \to \mathbb{N}$ is the initial marking function of a SPN;

(6) $M_0=[i]$, $[i]$ ($[o]$) represents only place i (o) contain one token;

(7) $\forall x \in P_I \cup T$, x is on a path from i to o and $\forall p \in P_{MI} \cup P_{MO}$. $\forall x \in P_{MI}$, ${}^\bullet x=\phi$ and $x^\bullet$ only includes a transition. $\forall x \in P_{Mo}$, $x^\bullet=\phi$ and ${}^\bullet x$ only includes a transition.

A composed Web service process composed of multiple Web services can be formally represented by a CSPN model.

Definition 4: Let $SPN_k=(P_k, T_k, F_k, M_{0k})$ be a *SPN* of the *k*-th cooperative service process net, $k = 1, 2, \ldots, n$. $CSPN = (P, T, F, M_0)$ is a composed Web service process net where

1) $P=\cup_{1\le k\le n}P_k$, $P=P_I \cup P_{MIO} \cup P_{CMI} \cup P_{CMO}$ are disjunct subsets of places, where $P_I=\cup_{1\le k\le n}P_{Ik}$, $P_{MIO}=(\cup_{1\le k\le n}P_{MOk})\cap(\cup_{1\le k\le n}P_{MIk})$ $P_{CMI}=\cup_{1\le k\le n}P_{MIk}\setminus\cup_{1\le k\le n}P_{MOk}$, $P_{CMO}=\cup_{1\le k\le n}P_{MOk}\setminus\cup_{1\le k\le n}P_{MIk}$;

2) $T = \cup_{1\le k\le n}T_k$;

3) $F = \cup_{1\le k\le n}F_k$;

4) M: $P\to\mathbb{N}$ is the marking function of a SPN;

4) M_0 is the initial marking, and $\forall p\in P_k$: $M_{0(p)}=M_{0k(p)}$, k= 1, 2, . . .,n. M_{ce} is the end marking where each end place *o* of SPN_k, k= 1, 2, . . .,n contains one token.

In a CSPN, $x\in P\cup T$ is called a node. Its preset ${}^{\bullet}x=\{y|(y,x)\in F\}$, and postset $x^{\bullet}=\{y|(x,y)\in F\}$.

Definition 5: Firing rules of a transition *t* in *CSPNs*. In a *CSPN*, $\forall t\in T$, *t* is enabled if $\forall p\in {}^{\bullet}t$; $M(p)\ge 1$. Firing enabled *t* results in a new marking M': $\forall p\in {}^{\bullet}t$: $M'(p)=M(p)-1$; $\forall p\in t^{\bullet}$: $M'(p)=M(p)+1$; $R(M_0)$ represents the set of all markings reachable from M_0.

Definition 6: A $CSPN = (P, T, F, M_0)$ is correct iff its corresponding composed Web service is executed properly.

Definition 7: If $CSPN = (P, T, F, M_0)$ is a CSPN composed of n SPNs: $SPN_k=(P_k, T_k, F_k, M_{0k})$, $k = 1, 2, \ldots, n$, then SPN_k is called the Web service process subnet of *CSPN*.

Definition 8: $ISPN = (IP, IT, IF, IM_0)$ is an internal net of a Web service process net $SPN = (P, T, F, M_0)$, where

(1) *IP* is a set of places. $IP=P_I$;

(2) *IP* includes two special internal places: *i* and *o*. Place *i* is a start place: ${}^{\bullet}i=\phi$. Place *o* is an end place: $o^{\bullet}=\phi$;

(3) $IT=T$;

(4) $IF=F\cap((P_I\times T)\cup(T\times P_I))$;

(5) M: $IP\to\mathbb{N}$ is the initial marking function of a ISPN; $IM_0=[i]$; $M_e=[o]$ is the end marking where the end place *o* contains one token.

Firing rules of transitions in ISPNs is the same as those in CSPNs.

3. Property Analysis of Composable Web Services

Suppose each input passing message is necessary for Web services. If a composed Web service can execute properly, then each input passing message of each Web service must be gotten. But each output passing message of each Web service does not have to be received by another Web service because output passing messages does not affect the proper execution of a composed Web service. So the below theorem can be obtained.

Theorem 1: If a composed Web service is executed properly, then its corresponding $CSPN = (P, T, F, M_0)$ satisfies: $P_{CMI}=\phi$.

Proof: If a composed Web service can execute properly, it means that every passing message of each Web service in the composed service is certain to be gotten. According to the definition of a CSPN, the input place set P_{CMI} of a CSPN must be empty, i.e. $P_{CMI}=\phi$.

Theorem 2: If a composed Web service is executed properly, then its corresponding $CSPN = (P, T, F, M_0)$ is deadlock-free except at the end marking M_{ce}.

Proof: According to definitions, A composed Web service can be executed properly if and only if each Web service in the composed Web service can be executed properly. A Web service is executed properly iff its input passing messages can be gotten and it can terminate normally. From the two definitions, it can be inferred that if a composed Web service is executed properly, then each Web service in the composed service can terminate normally.

Proof by contradiction is used. Suppose the *CSPN* is deadlock. It can be inferred that at least one of Web services in the composed Web service is not executed properly. However, this conclusion is contradictory to the above inference from prerequisites. Hence, the theorem 2 is correct.

Definition 9: C is a path leading from node n_1 to n_k iff $\exists$ a node sequence $<n_1, n_2, ..., n_k>$ in CSPNs such that $(n_i, n_{i+1})\in F$, i=1, 2, ..., k-1. Let &(C) represent the alphabet of a path C, i.e., $\&(C)=\{n_1, n_2, ..., n_k\}$. C is an elementary path iff $\forall n_i, n_j\in\&(C)$, if $i\ne j$, $n_i\ne n_j$. Node n_j is an inheritor

of node n_i in C (notation $n_i \prec_C n_j$) iff $\exists$ n_i, n_{i+1}, ..., $n_j \in$&(C) such that (n_i, n_{i+1}), (n_{i+1}, n_{i+2}), ..., (n_{j-1}, n_j)$\in$F.

In our study, the passing message places in CSPNs are divided into two classes. Places standing for input passing messages are denoted by P_{CMI} and places standing for output passing messages are denoted by P_{CMO}. The sending transition of place *p* is denoted by s(*p*). The receiving transition of place *p* is denoted by r(*p*).

Theorem 3: Let a CSPN be composed of two deadlock-free Web service process subnets: SPN_1 and SPN_2 and $P_{CMI}=\phi$. $ISPN_1$ and $ISPN_2$ are deadlock-free except at their end marking. For any elementary path C_j leading from *i* to *o* in SPN_j, j=1, 2, and l$\in${0,1}, there exists no pair of passing message places *p* and *q* such that $r_{l+1}(p) \prec_{C_{l+1}} s_{l+1}(q)$ and $r_{2-l}(q) \prec_{C_{2-l}} s_{2-l}(p)$ are simultaneously satisfied if the CSPN is deadlock-free except at M_{ce}.

Proof: In the proof, reduction to absurdity is used. Suppose the conclusion is not correct, i.e., for some elementary path C_j leading from *i* to *o* in SPN_j, j=1, 2, and l$\in${0,1}, there exists some pair of passing message places *p* and *q* such that $r_{l+1}(p) \prec_{C_{l+1}} s_{l+1}(q)$ and $r_{2-l}(q) \prec_{C_{2-l}} s_{2-l}(p)$ are simultaneously satisfied. That indicates that there exists the case such that SPN_{l+1} waits to receive the passing message of *p* from SPN_{2-l} and SPN_{2-l} waits to receive the passing message of *q* from SPN_{l+1}, which is called circular deadlock. In this case, $s_{l+1}(q)$ and $s_{2-l}(p)$ are two dead transitions, and the CSPN is deadlock. But this conclusion is contradictory to the condition in the theorem 3 that CSPN is deadlock-free except at M_{ce}. Thereby, the supposition is not true and the Lemma is correct.

Extend **theorem 3** to a general case involving multiple Web service process subnets.

Theorem 4: Let a CSPN be composed of n Web service process subnets: SPN_i, i =1, ..., n, $P_{CMI}=\phi$ and $ISPN_i$ is deadlock-free except at their end marking. $\forall u,v \in \{1, ..., n\}$, $u \neq v$, any two elementary paths C_u and C_v leading from *i* to *o* in SPN_u and SPN_v, respectively, there exists no pair of passing message places *p* and *q* in P_{MIO} such that for k, l$\in${u, v}, k$\neq$l, $r_k(p) \prec_{C_k} s_k(q)$ and $r_l(q) \prec_{C_l} s_l(p)$ are simultaneously satisfied if the CSPN is deadlock-free except at M_{ce}.

Proof: It is similar to the **theorem 3**. Based on **theorem 3**, it is easy to verify the **Theorem 4**.

From **theorem 3** and **theorem 4**, although the deadlock of a CSPN depends on its dynamic execution, it is easy to verify that a CSPN is deadlock based on the net structure, thus the corresponding composed Web service is not executed properly based on **theorem 4**.

For the $CSPN_1$ in Figure 1, SPN_1 and SPN_2 are the two corresponding Web service process subnets of the $CSPN_1$. In the following, we apply **theorem 3 and theorem 4** to analyze the deadlock-freedom of the $CSPN_1$. The deadlock-freedom of $ISPN_1$ and $ISPN_2$ is obvious except at their end marking. P_{MIO}= {p_6, p_7, p_8, p_9}, s(p_6)= t_7, r (p_6)= t_2 , s(p_7)= t_2, r(p_7)= t_8, s(p_8)= t_3, r (p_8)= t_6 ,s (p_9)= t_6 and r(p_9)= t_4. In SPN_1, there is one elementary path leading *i* to *o*, C_1= $i_1 \rightarrow t_1 \rightarrow p_2 \rightarrow t_2 \rightarrow p_3 \rightarrow t_3 \rightarrow p_4 \rightarrow t_4 \rightarrow o_1$. In PS_1, there is one elementary path, i.e., C_2= $i_2 \rightarrow t_5 \rightarrow p_{11} \rightarrow t_6 \rightarrow p_{12} \rightarrow t_7 \rightarrow p_{13} \rightarrow t_8 \rightarrow o_2$. It is easy to verify that $\forall$p, q$\in$ P_{MIO}, there exists a pair of interactive interface places p_6 and p_8 such that r(p_6)= $t_2 \prec_{C_1}$ s(p_8)= t_3 and r(p_8)= $t_6 \prec_{C_2}$ s(p_6)= t_7 are simultaneously satisfied. According to **theorem 3 and theorem 4**, the $CSPN_1$ is not deadlock-free, thus the corresponding composed Web service cannot be executed properly.

It is a necessary condition of keeping deadlock-free of a CSPN composed of n deadlock-free Web service process subnets that there is no circular deadlock except at their end markings, but not a sufficient condition. For example, the CSPN composed of two Web service process subnets is shown in Figurre 2. For the $CSPN_2$ in Fig.2, SPN_3 and SPN_4 are the two corresponding Web service process subnets of the $CSPN_2$. The deadlock-freedom of $ISPN_3$ and $ISPN_4$ is easy to prove. P_{MIO}={p_8, p_9, p_{10} },s(p_8)= t_4, r (p_8)= t_2 , s (p_9)= t_3, r (p_9)= t_5 , s(p_{10})= t_1, r(p_{10})= t_6. In SPN_3, there are two elementary paths leading from *i* to *o*. $C_3^{(1)}$=$i_1 \rightarrow t_1 \rightarrow p_3 \rightarrow t_7 \rightarrow o_1$. $C_3^{(2)}$=$i_1 \rightarrow t_2 \rightarrow p_2 \rightarrow t_3 \rightarrow p_3 \rightarrow t_7 \rightarrow o_1$. In SPN_3, there is one elementary path, i.e., $C_4^{(1)}$=$i_2 \rightarrow t_4 \rightarrow p_5 \rightarrow t_5 \rightarrow p_6 \rightarrow t_6 \rightarrow o_2$. It is easy to verify that $\forall$p, q$\in P_{MIO}$, there does not evidently exist j$\in${1, 2} and k$\in${1} such that each of ($r_3(p) \mathrm{p}_{C_3^{(j)}} s_3(q)$) $\wedge$ ($r_4(q) \mathrm{p}_{C_4^{(k)}} s_4(p)$) and ($r_4(p) \mathrm{p}_{C_4^{(k)}} s_4(q)$) $\wedge$($r_3(q) \mathrm{p}_{C_3^{(j)}}$

$s_3(p)$) is satisfied. It can be verified that the $CSPN_2$ in Figure 2 is not deadlock-free except at its end marking.

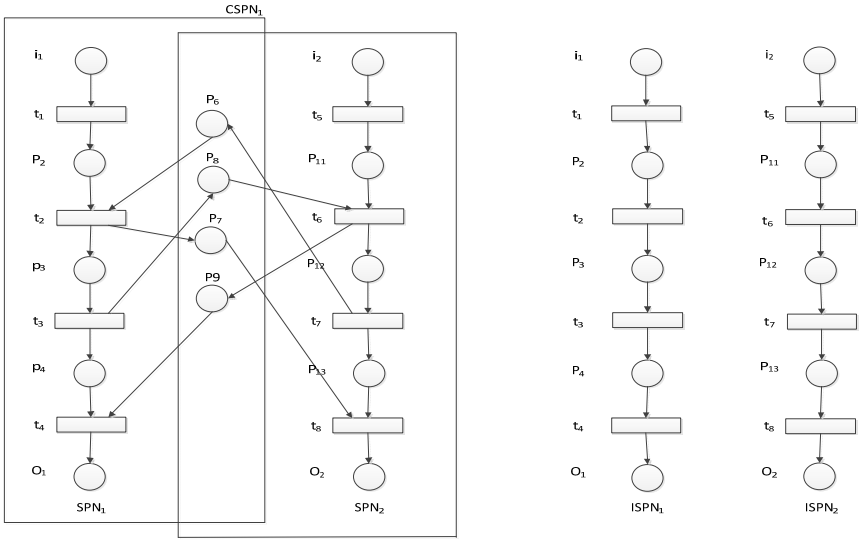

Figure 1. The $CSPN_1$ composed of SPN_1 and SPN_2 and their corresponding $ISPN_1$ and $ISPN_2$

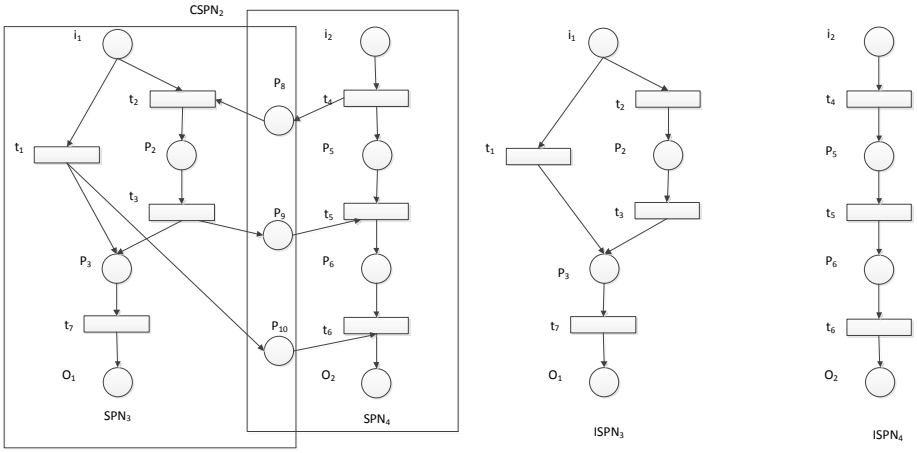

Figure 2. The $CSPN_2$ composed of SPN_3 and SPN_4 and their corresponding $ISPN_3$ and $ISPN_4$

Generally, the deadlock-freedom of a CSPN depends on not only the order of sending and receiving actions among process subnets, but also the choice of firing sequences in conditional routing constructs.

Although it is a necessary condition of keeping deadlock-free of a CSPN composed of n deadlock-free Web service process subnets at their end marking that there is no circular deadlock, not a sufficient condition, for some interesting subclasses of CSPNs, it is sufficient and necessary condition. We focus on a class of CSPNs in which none of the sending and receiving actions is included in any conditional routing construction.

Definition 10: A Web service process subnet *SPN* is CC- limited iff $\forall\ p_1, p_2 \in P_I$, if there exist two elementary paths C_1 and C_2 ($C_1 \neq C_2$) leading from *i* to *o*, and $\&(C_1) \cap \&(C_2) = \{p_1, p_2\}$, then $\forall t \in T \cap ({}^{\bullet}P_{MO} \cup {}^{\bullet}P_{MI})$: $t \notin \&(C_1) \cup \&(C_2)$. A *CSPN* is CC-limited iff each of its Web service process subnets is CC-limited.

Theorem 5: Let a CSPN be CC- limited and $P_{CMI}=\phi$, which is composed of two deadlock-free Web service process subnets except at their end markings: SPN_1 and SPN_2. For any elementary path C_j leading from *i* to *o* in SPN_j, j=1, 2, and $l\in\{0,1\}$, there exists no pair of passing message places *p* and *q* such that $r_{l+1}(p)\prec_{C_{l+1}} s_{l+1}(q)$ and $r_{2-l}(q)\prec_{C_{2-l}} s_{2-l}(p)$ are simultaneously satisfied iff the CSPN is deadlock-free except at its end marking.

Sufficiency: Since both SPN_1 and SPN_2 are deadlock-free except at their end markings, the deadlock-freedom except at its end marking of the CSPN depends on the deadlock of the transitions related to the places in P_{MIO} and their precedence order in each Web service process subnet. However, the order cannot be updated when a CSPN is constructed, i.e. the structural order of the transitions in every Web service process subnet is the same as that in the CSPN. Consequently, when both Web service process subnets are deadlock-free in a CSPN, if the CSPN is not deadlock-free, this means that the incorrect order of the transitions related to the passing message places in some Web service process subnets leads to some dead transitions in the CSPN.

Because the CSPN is CC-limited, this means that the sending and receiving passing message places between SPN_1 and SPN_2 are only in sequential routing constructions in SPN_1 and SPN_2. By means of the conditions of the theorem, any elementary path C_j leading from *i* to *o* in SPN_j, j=1, 2, and $l\in\{0,1\}$, there exists no pair of passing message places *p* and *q* such that $r_{l+1}(p)\prec_{C_{l+1}} s_{l+1}(q)$ and $r_{2-l}(q)\prec_{C_{2-l}} s_{2-l}(p)$ are simultaneously satisfied. That is to say, there exists no case such that SPN_{l+1} waits to receive the message of *p* from SPN_{2-l} and SPN_{2-l} waits to receive the message of *q* from SPN_{l+1}. Thereby, each transition in SPN_1 and SPN_2 is deadlock-free in the CSPN and the CSPN is deadlock-free.

Necessity: CC-limited CSPN are the special case of CSPNs. So the conclusion is correct based on **theorem 4.**

Extend **Theorem 5** to a general case involving multiple Web service process subnets

Theorem 6: Let a CSPN be CC-limited and $P_{CMI}=\phi$, which is composed of n deadlock-free Web service process subnets at their end markings: SPN_i, i =1, ..., n. If $\forall u, v\in\{1, ..., n\}$, $u\neq v$, $(P_{MIu}\cup P_{MOu})\cap(P_{MIv}\cup P_{MOv})\neq\phi$, and any two elementary paths C_u and C_v leading from *i* to *o* in SPN_u and SPN_v, respectively, there exists no pair of passing message places *p* and *q* in $P_{Iu}\cap P_{Iv}$ such that for k, $l\in\{u, v\}$, $k\neq l$, $r_k(p)\prec_{C_k} s_k(q)$ and $r_l(q)\prec_{C_l} s_l(p)$ are simultaneously satisfied iff the CSPN is deadlock-free except at its end marking.

Proof: It is similar to the **Theorem 5**. Based on **Theorem 5**, it is easy to verify the theorem 6.

According to **Theorem 5 and Theorem 6**, a rigorous analysis approach same as **theorem 3 and theorem 4** is given based on only static net structures of CSPNs to verify the liveness preservation for CC-limited CSPNs.

4. Conclusion

The application of SOC reduces the complexity of business integration within an enterprise and across organizations. To study the satisfied properties of composable Web services, formal models based on Petri nets are proposed in this paper: the Web service process nets (SPNs) representing Web services, the internal service process nets (ISPNs) representing the inner Web service process and the composed Web service process nets (CSPNs) denoting the a composed Web service. Based on the above three formal models, the properties of composable Web services are studied. The relationship between the proper execution of Web service composition and interface passing messages and deadlock of composition workflow is discussed.If a composed Web service is executed properly, its corresponding CSPN is deadlock-free except at the end marking. To reduce complexity of analysis, the relation between a CSPN and its corresponding Web service process subnets are discussed. If the CSPN composed of multiple deadlock-free Web service process subnets is deadlock-free except their end markings, no circular deadlocks exist. Although the deadlock-freedom is a dynamic property, the property of deadlock-freedom can be judged based on static net structures by means of the proposed methods. However, it is a necessary condition. A special subclass is come up with in the paper, it is a sufficient and necessary condition for CC-limited CSPN. Petri nets are also adopted in some papers to model Web service. Different from

previous methods, analysis methods presented in this paper are based on the static structure of CSPNs.By these methods, it is easier to analyze properties of composable Web services.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under grant 61472228, 61170078; the Natural Science Foundation of Shandong province under grant ZR2014FM009, the doctoral program of higher education of the specialized research fund of China under grant 20113718110004; the Project of Shandong Province Higher Educational Science and Technology Program under Grant number J12LN11; the China's Post-doctoral Science Fund under grant 2012M521362;the Project of Shandong Post-doctoral Fund under grant 201303071;the international cooperation training Project of Shandong Province Higher Educational outstanding youth backbone teachers; and Basic Research Program of Qingdao City of China under grant No. 13-1-4-116-jch.

References

[1] Liu J, Luo XJ, Li BN, et al. An Intelligent Job Scheduling System for Web Service in Cloud Computing. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(6): 2956-2961.
[2] Liu W, Du YY, Yan C. A service selection method based on Web service Clusters. *Journal of Applied Science*. 2013; 13: 5734-5738.
[3] Maamar Z, Wives LK, Al-Khatib G, Sheng QZ, et al. *From communities of web services to marts of composite web serivces*. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and applications. Perth, WA. 2010: 958-965.
[4] Liu W, Du YY, Yan C. A web service discovery and composition method based on service classes. *Journal of Software Engineering*. 2013; 7: 68-76.
[5] Li H, Wang HY, Cui LZ. *Automatic composition of web services based on rules and meta-services*. Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design. Melbourne, Vic. 2007: 496-501.
[6] Liu W, Du YY, Yan C. A method to calculate recommendation trust of Web services. *Journal of Software Engineering*. 2014; 8: 108-115.
[7] Argentina BA. *A model-driven approach to describe and predict the performance of composite services*. Proceedings of the 6th International Workshop on Software and Performance. New York. 2007: 78-89.
[8] Liu W, Du YY, Yan C. Soundness preservation in composed logical time workflow nets. *Enterprise Information System*. 2012; 6: 95-113.
[9] Liu W, Du YY, Zhou MC, Yan C. Transformation of Logical Workflow Nets. *IEEE Transactions on Systems Man and Cybernetics: Systems*. 2014; 44(10): 1401-1412.
[10] Azgomi MA, Entezari-Maleki R. Task scheduling modelling and reliability evaluation of grid services using coloured Petri nets. *Future Generation Computer Systems*. 2010; 26: 1141-1150.
[11] Charfi A, Schmeling B, Mezini M. *Reliable messaging for BPEL processes*. IEEE International Conference on Web Services. Chicago. 2006: 293-302.
[12] Weerawarana S, Curbera F, Leymann F, Storey T, Ferguson DF. Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more. Prentice Hall PTR. 2005.
[13] Tang XF, Jiang CJ, Zhou MC. Automatic Web service composition based on Horn clauses and Petri nets. *Expert Systems with Applications*. 2011; 38(10): 13024-13031.
[14] Cambronero ME, Díaz G, Valero V, Martínez E. Validation and verification of web services choreographies by using timed automata. *The Journal of Logic and Algebraic Programming*. 2011; 80(1): 25-49.
[15] Wang YM. Research on Web Service Composition Algorithm Using Description Logic. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(1): 852-858.
[16] Yun BS. Formal Modeling of Trust Web Service Composition Using Picalculus. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(8): 4385-4392.