ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2009/03(02)-0173-15
DOI:10.3778/j.issn.1673-9418.2009.02.006

E-mail:fcst@public2.bta.net.cn
http://www.ceaj.org
Tel:+86-10-51616056

Modeling and Verifying Composite Web Services Based on Semantic Annotated Petri Nets*

LEI Lihui[1,2], DUAN Zhenhua[1+]

1. Institute of Computing Theory and Technology, Xidian University, Xi'an 710071, China

2. State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China

+ Corresponding author: E-mail: zhhduan@mail.xidian.edu.cn

语义标记 Petri 网的组合 Web 服务建模与验证 *

雷丽晖[1,2],段振华[1+]

1. 西安电子科技大学 计算理论与技术研究所,西安 710071

2. 武汉大学 软件工程国家重点实验室,武汉 430072

摘 要:随着 Web 服务组合的发展,整合业务过程成为可能。组合 Web 服务可以被看作是基于过程的工作流。由于死锁、不安全和不可达等流的设计错误会影响组合 Web 服务的有效执行,因此这些错误应在组合 Web 服务执行前被检测出并修改。提出了基于语义标记 Petri 网的组合 Web 服务建模与验证方法。首先提出语义标记 Petri 网(SaPNs),并给出其语义;用受限描述逻辑 tableau 算法获得组合 Web 服务;使用 SaPNs 描述组合 Web 服务及其组成部分;最后,使用基于 SaPNs 的分析方法验证了组合 Web 服务。使用该方法在开放的 Internet 环境下可以获得满足客户需求的、可靠的组合 Web 服务。

关键词: Petri 网;Web 服务;语义匹配

文献标识码:A **中图分类号:**TP393

LEI Lihui, DUAN Zhenhua. Modeling and verifying composite Web services based on semantic annotated Petri Nets. Journal of Frontiers of Computer Science and Technology, 2009,3(2):173-187.

* The National Natural Science Foundation of China under Grant No.60433010,60873018 (国家自然科学基金); the Defense Pre-Research Project of China under Grant No.51315050105 (装备预先研究项目); the Specialized Research Foundation for the Doctoral Program of Chinese Higher Education under Grant No. 200807010012 (高等院校博士学科点专项科研基金项目); the Foundation of State Key Laboratory of Software Engineering of Wuhan University No.20080713 (武汉大学软件工程国家重点实验室开放基金项目).
Received 2008-07, Accepted 2008-09.

Abstract: Integrating business processes becomes practicable along with the development of Web services composition. Composite Web services can be regarded as process-based workflows. Since error design of flow structures, such as deadlock, unsafeness, non-reachability and so on, will affect composite Web services performance, these errors should be detected and corrected before composite Web services are executed. An approach for modeling and verifying composite Web services based on semantic annotated Petri Nets is proposed. Firstly, semantic annotated Petri Nets (SaPNs) is presented and the semantics of SaPNs is given. Secondly, a restricted description logic tableau algorithm is utilized to obtain composite Web services. Thirdly, SaPNs is used to represent a composite Web service and the participating services that are invoked by this composite Web service. Finally, the composite Web services represented with the SaPN are verified by means of some analysis methods based on SaPNs. With this approach, a reliable composite Web services meeting client requirements in an open Internet environment can be achieved.

Key words: Petri Nets; Web services; semantic matching

1 Introduction

Web services can be viewed as universally accessible components deployed over the Internet. They are designed to support interoperable machine-to-machine interaction over a network[1]. Considerable efforts have been spent to define standards and technologies for Web services, e.g., SOAP[2] for services communication, WSDL[3] for services interface description, UDDI[4] for services publication and discovery, and BPEL4WS[5] for services interaction model description and so on. Since all of them are based on XML-centric approaches, Web services have the capability of implementing distributed applications across not only platforms but also program languages.

Nowadays, integrating business processes becomes practicable along with the development of Web services composition. To do so, individual business processes are wrapped by Web services, and then these individual business processes are integrated by means of composing Web services. In this paper, the procedure of composing Web services is called Web services composition and the result of composing Web services is called composite Web service. Besides, the selected services for integrating business processes are called candidate services and a participating service is one candidate service which is executed at integrated business processes runtime. In a sense, integrated business processes are created with composite Web services.

A composite Web service can be regarded as a process-based workflow. It is worth noting that some error designs of flow structures, such as deadlock, unsafeness and non-reachability, will affect composite Web services performance.

(1) Deadlock. During the execution of a composite Web service, if there is a participating service that is impossible to receive the input so that the composite Web service cannot be performed continuously, then a deadlock problem of the composite Web service occurs. Obviously, the execution of a composite Web service with deadlock cannot terminate normally, which makes the integrated business processes broken down.

(2) Unsafeness. During the execution of a composite Web service, if a participating service sends an output (i.e., a message), however, the output makes no sense to the composite Web service, i.e., it cannot be consumed by other participating services, then an unsafeness problem of the composite Web service occurs. The more instances of the composite Web service are creat-

ed, the more useless messages are stored in a channel. At last, the channel overflows.

(3) Non-Reachability. A composite Web service defines the temporal sequences by which participating services are executed. If an instance of the composite Web service is executed by one temporal sequence, it should produce a client expected result. If for any client expected results, a temporal sequence can be found so that an instance of the composite Web service executed by it can produce the expected result, the composite Web service is reachable; otherwise a non-reachability problem of the composite Web service occurs.

Obviously, these errors should be detected and corrected before composite Web services are executed. Nowadays, many researchers in both industry and academia are making efforts to find the methods to model and verify composite Web services. Niels Lohmann et al. in [6] presented a method to verify global interactions between business processes of different partners based on Petri Nets. In [7] Yu Huang et al. proposed an extended Petri Net that was used to model the flow of messages exchanged by a Web service which participated in choreographed interactions with other services. Kang Hui et al. in [8] proposed a CPN-based model for Web services composition, described how to translate WS4BPEL to CPN model, and discussed the analysis and verification of services composition. Jan Hidders et al. in [9] introduced an extended Petri Net which is a formal, graphical workflow language for modeling and analyzing the dataflows.

Using Petri Nets and their extensions to model inter-organizational business processes, only syntactic composition of distributed business processes can be solved. However, the missing semantic representation of Petri Nets can hamper the inter-connectivity of business processes, since different business partners have their own specific vocabularies, even if they share a similar requirement.

The semantic annotated Petri Nets (SaPNs) are proposed in this paper. Modeling Web services with SaPNs, the semantics of exchanged data between the services can be dealt with so that different business processes with their own specific vocabularies can be connected. Firstly, SaPNs is used to represent a composite Web service and the participating services; then the interactions between those services are verified by means of some analysis methods based on SaPNs. With this approach, a reliable composite Web services meeting client requirements in an open Internet environment can be received.

In the rest of the paper, section 2 gives a motivation example for discussion. Section 3 introduces SaPNs and the semantics of SaPNs. Section 4 illustrates how to use the restricted description logic tableau algorithm to obtain a composite Web service. Section 5 describes how to represent a composite Web service and the participating services with SaPNs. Section 6 describes the formal methods to verify the interactions between services. Conclusions are drawn in section 7.

2 Motivation Example

Suppose a client wants to find such a Web service that can help it purchase a book and a CD. This purchasing procedure is depicted as follows. Firstly, the client sends *Bookname* to the service and receives *Bookbuyinfo* from the service if the book is in the stock, or *Boutstock* if the book is out of stock. Similarly, the client also sends *CDname* to the service and receives *CDbuyinfo* from the service if the CD is in the stock, or *Coutstock* if the CD is out of stock. Since the above two activities have no causal relation, they are executed concurrently. If the client receives both *Boutstock* and *Coutstock*, the service instance terminates, otherwise the client sends *Creditcard* and *Goodsbuyinfo*

(*Bookbuyinfo* and/or *CDbuyinfo*) to the service. If it receives *Forbiddance* returned from the service, then this service instance terminates, otherwise it receives *Authorization*, *Payinfo* and the service instance goes on being executed: the client sends *Payinfo* and *Cargotype* to the service for shipping support and receives *Order* returned from the service. Finally, the service instance terminates. The above italics are parameters that should be processed by the service.

If there does not exist such a Web service, a composite Web service can be constructed to satisfy the client requirements. To obtain such a composite Web service, we should do as follows: (1) finding several candidate services according to the parameters processed by the service; (2) composing the candidate services found in the previous step. Note that the output of the client is the input of the service; the input of the client is the output of the required service; moreover, the data always flows from the "output" to the "input". For the above example, the composite Web service is intuitively depicted in Fig.1.

3 Semantic Annotated Petri Nets

To model and analyze Web services, this Section introduces semantic annotated Petri Nets and gives the semantics of SaPNs.

Definition 1 A semantic annotated Petri Net is a 6-tuples $SN=(P,T,F,\Sigma,M_0;a)$, where:

(1) P is a finite set of control places;

(2) T is a finite set of transitions and $P\cap T=\phi$;

(3) $F\subseteq(P\times T)\cup(T\times P)\cup(\Sigma\times T)\cup(T\times\Sigma)$ is a finite set of arcs;

(4) Σ is a finite set of messages. Each element of Σ is 4-tuples (I,O,S,R), where I and O are two finite sets of letters, S and R are two letters;

(5) $M=(M_c,M_d)$ is a marking function, $M_c:P\rightarrow\{0,1\}$, $M_d:\Sigma\rightarrow\{0,1\}$, and $M_0=(Mc_0,Md_0)$ is the initial marking;

(6) a: $T\rightarrow\{\Sigma\times\text{dir}\}\cup\{\tau\}$, where dir=$\{+,-\}$.

A SaPN is represented by a directed graph with three node types called places, transitions and messages. Places and transitions are connected via solid directed arcs, while transitions and messages are connected with dotted directed arcs. Connections between two nodes with the same type are not allowed. Further more, places and messages cannot be connected. Places are represented by solid circles, transitions by rectangles and messages by dotted circles.

Three kinds of transitions are depicted in Fig.2. Each transition is annotated by a letter τ or a binary-tuple $((I,O,S,R),d)$, where $(I,O,S,R)\in\Sigma$ represents a message and d is a symbol, either "+" or "-". Not more than one message is connected graphically to a transition through a dotted arc. If the transition t is annotated by $((I,O,S,R),+)$, the dotted arc is from the transition t to the message (I,O,S,R), while if the tran-

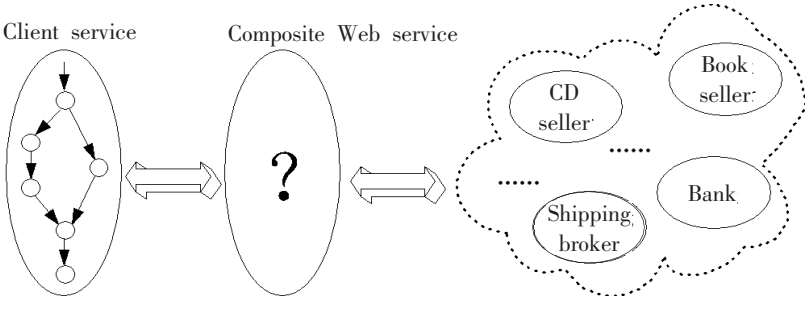


Fig.1 The interactions between the composite Web service and the client

图 1 组合 Web 服务与用户之间的交互

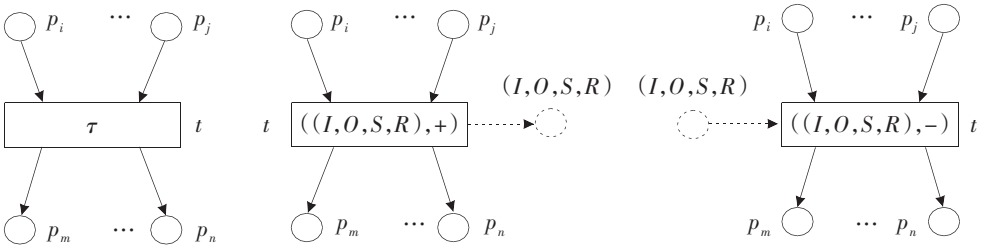


Fig.2 Three kinds of transitions in SaFNs

图 2 SaFNs 中的三种迁移

sition t is annotated by $((I,O,S,R),-)$, the dotted arc is from the message (I,O,S,R) to the transition t, otherwise the transition t is annotated by the letter τ and there is no messages connected with this transition. In a SaPN, tokens in the control places are called control conditions and tokens in the messages are called data conditions.

The enabling conditions and firing rules of transitions are defined in the following. $R(M_0)$ denotes a set of all markings reachable from M_0.

Definition 2 Let $SN=(P,T,F,\Sigma,M_0,a)$ be a semantic annotated Petri Net, $M=(M_c,M_d)\in R(M_0)$. $\forall t\in T$, t is enabled at M if and only if:

(1) $a(t)=\tau$ and $\forall p\in {}^{\bullet}t: M_c(p)=1$;

(2) $a(t)=(m,+)$ where $m\in\Sigma$, further, $\forall p\in {}^{\bullet}t: M_c(p)=1$;

(3) $a(t)=(m,-)$ where $m\in\Sigma$, further, $\forall p\in {}^{\bullet}t: M_c(p)=1$ and $M_d(m)=1$.

According to the above Definition, if a transition is enabled at $M=(M_c,M_d)$, its control conditions and data condition must be satisfied.

Definition 3 Let $SN=(P,T,F,\Sigma,M_0,a)$ be a semantic annotated Petri Net, $M=(M_c,M_d)\in R(M_0)$. $\forall t\in T$, t is enabled at M, then it may fire and its firing generates a new marking M', i.e., $M[t>M'=(M_c',M_d')$, where:

(1) $M_c'(p)=M_c(p)+1$, if $p\in t^{\bullet}-{}^{\bullet}t$;

(2) $M_c'(p)=M_c(p)-1$, if $p\in {}^{\bullet}t-t^{\bullet}$;

(3) $M_c'(p)=M_c(p)$, if $p\notin(t^{\bullet}-{}^{\bullet}t)\cup({}^{\bullet}t-t^{\bullet})$;

(4) if $a(t)=(m,+)$, $m\in\Sigma$, then $M_d'(m)=M_d(m)+1$, and for all $\widetilde{m}\in\Sigma-\{m\}$, $M_d'(m)=M_d(\widetilde{m})$;

(5) if $a(t)=(m,-)$, $m\in\Sigma$, then $M_d'(m)=M_d(m)-1$, and for all $\widetilde{m}\in\Sigma-\{m\}$, $M_d'(m)=M_d(\widetilde{m})$;

(6) if $a(t)=\tau$, then for all $m\in\Sigma$, $M_d'(m)=M_d(m)$.

To represent a Web service with a SaPN, all exchanged messages between the service and its client constitute the set Σ. A message is represented by a 4-tuple (I,O,S,R). I is a finite set of letters and each element of I represents an input parameter (referring to a concept in an ontology) received by the service. O is also a finite set of letters and each element of O represents an output parameter (referring to a concept in an ontology) sent by the service. Moreover, I and O cannot be empty at the same time. S is a letter and represents the sender of the message, while R is a letter and represents the receiver of the message. Note that, a service may not only invoke its client, but also be invoked by its client.

Therefore, data can flow either from the service to the client or from the client to the service. The following example is given to illustrate this scenario. Suppose there are services S_1 and its client C_1. Service S_1 is modeled by a SaPN. It worths noting that the message (I,O,S_1,C_1) means that the service S_1 sends the out-

put O to the client C_1 and then receives the input I returned from the client C_1, while the message (I,O,C_1,S_1) means that the service S_1 receives the input I sent by the client C_1 and then returns the output O to the client C_1. Furthermore, in a SaPN representing a Web service, for a $p \in P$, $M_c(p)=1$ means that the place p has a token, i.e., the control condition is true, while $M_c(p)=0$ means that the place p has no token, i.e., the control condition is false. For a $m \in \Sigma$, $M_d(m)=1$ means that the message m has a token, i.e., the data condition is true (that is to say, the message has been produced by the message sender can be consumed by the message receiver), while $M_d(m)=0$ means that the message m has no token, i.e., the data condition is false (that is to say, the message does not exist because it has not been produced or has been consumed by the message receiver).

Moreover, transitions in a SaPN are used to describe actions that the service do. Each transition is annotated by a letter τ or a binary-tuples $((I,O,S,R),d)$, where (I,O,S,R) represents a message and d is a symbol, either "+" or "-". As shown in the above example, modeling service S_1 with SaPNs, the action that service S_1 receives input I sent by client C_1 and then returns output O to client C_1 is represented by $((I,O,C_1,S_1),-)$, while the action that service S_1 sends output O to client C_1 and then receives input I returned from client C_1 is represented by $((I,O,S_1,C_1),+)$. Other actions that the service evaluates conditions to decide what to do next are represented by letter τ.

Definition 4 A SaNP $SN=(P,T,F,\Sigma,M_0,a)$ represents a Web service iff:

(1) there exists $p_i \in P$ and $\overset{\bullet}{p_i}=\phi$;

(2) there exists $p_{o1},p_{o2},\cdots,p_{ok} \in P$ and $\forall j \in \{1,2,\cdots,k\}$, $p_{oj}^{\bullet}=\phi$;

(3) for all $(I,O,S,R) \in \Sigma$.

S and R cannot be the same;

I and O cannot be empty at the same time;

Each element of I and O refers a concept of an ontology;

(4) for all transitions in T, if its annotation is:

$((I,O,S,R),+)$, S is the identifier of this service and the exchanged data first flows out of the service and then flows into the service;

$((I,O,S,R);-)$, R is the identifier of this service and the exchanged data first flows into the service and then flows out of the service;

τ, there is no data exchanging between the service and its client.

4 Composing Web Services

In this section we show how to solve the problem of composing Web services both in the general case and in the context of the above example. Suppose that there exists a set of existing Web services and each service has a description in OWL-S[10]. Moreover, there does not exist a service that satisfies the client requirements.

4.1 Finding Candidate Services

OWL-S is both an ontology and a language used to describe semantic Web services. Being a part of OWL-S, the process model describes the interaction model between a service and its clients. The OWL-S process model is organized as a process-based workflow. Each atomic process is described by three components: inputs, preconditions and results that describe outputs and effects produced by the process under different conditions. An atomic process might have two results (e.g., the process for goods payment has two results, *authorization* and *forbiddance*); however, only one of results is produced when the process is executed completely (*authorization* and *forbiddance* cannot be returned at the same time and which one can be returned

depending on whether or not the credit card is valid). In terms of composing Web services, we only need to consider the functions of the atomic processes, therefore the preconditions and effects of the atomic processes cannot be considered here.

With semantic matching[11-12], we search those atomic processes whose input and/or output contains some or all of the parameters processed by the required service (i.e., the composite Web service constructed to meet the client requirements). Those services contain one or some of the selected atomic processes are candidate services. These selected atomic processes can be represented by description logic (DL) formulas and semantic annotated Petri Nets.

Suppose there is an atomic process that receives input $(I_1,\cdots,I_n)$ and returns output $(O_i,\cdots,O_j)$ or $(O_k,\cdots,O_h)$. $(O_i,\cdots,O_j)$ represents the results that the client expects to obtain and $(O_k,\cdots,O_h)$ represents the results that the client does not expect to receive. The expected and unexpected results cannot be returned to client at the same time. This atomic process can be represented by the formula and the SaPN in Fig.3.

For the description logic formula, it is a disjunction and two items cannot be true at the same time; moreover, I and O with subscripts are atomic propositions representing the valid input parameter I and output parameter O of atomic processes. For the SaPN, it has two transitions that have the same input place called initial places; the two output places of the transitions are called end places. If the token is in the initial place of the SaPN, it implies that the atomic process receives the valid input; if the token is in the end places of the SaPN, it implies that the atomic process returns the valid output (expected output or unexpected output).

4.2 Composing Candidate Services

For ease of exposition, a simpler example is given to illustrate our approach to composing candidate services. Suppose there are three candidate services and each has only one atomic process. These atomic processes are denoted by P_1, P_2 and P_3. Here, I and O with subscripts are atomic propositions representing the valid input and output of services. P_1 has the input I_1 and I_4 and output O_1; P_2 has the input I_2 and I_5 and output O_2; P_3 has the input I_2 and I_3 and output O_{I_4} and O_{I_5}; moreover, O_{I_4} equals I_4 and O_{I_5} equals I_5. Suppose the client requires that the required service has the input I_1, I_2 and I_3 and output O_1 and O_2. They are represented in Fig.4.

A restricted description logic tableau algorithm is employed to solve the problem of composing atomic processes. The result of the automatic reasoning procedure with this restricted tableau algorithm is depicted in Fig.4. First, let $I=I_1\wedge I_2\wedge I_3$, which describes the input of the required service. Then, we build those formulas P_1, P_2 and P_3. Next, the formulas are trans-

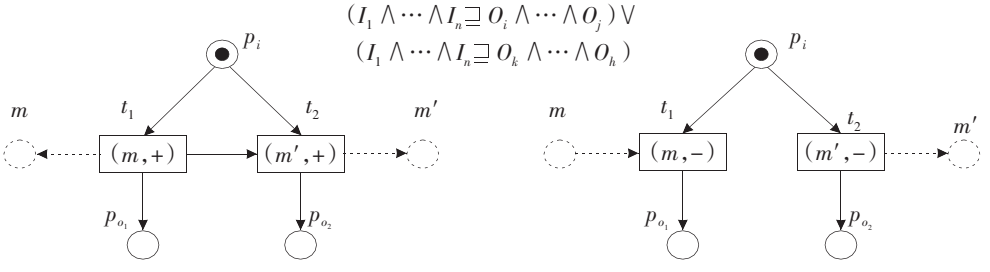


Fig.3 Two representations of an atomic process

图 3 原子过程的两种描述

$P_1=I_1 \wedge I_4 \sqsupseteq O_1$
$P_2=I_2 \wedge I_5 \sqsupseteq O_2$
$P_3=I_2 \wedge I_3 \sqsupseteq O_{I_4} \wedge O_{I_5}$
$(O_{I_4}=I_4, O_{I_5}=I_5)$

$S_1=\neg I_1 \vee \neg I_4 \vee O_1$
$S_2=\neg I_2 \vee \neg I_5 \vee O_2$
$S_3=\neg I_2 \vee \neg I_3 \vee (I_4 \wedge I_5)$
$I=I_1 \wedge I_2 \wedge I_3$

$K_0: S_1 \wedge S_2 \wedge S_3 \wedge I$
$K_1: I_4 \wedge I_5 \wedge S_1 \wedge S_2 \wedge I$
$K_2: O_1 \wedge I_4 \wedge I_5 \wedge S_2 \wedge I$
$K_3: O_2 \wedge I_4 \wedge I_5 \wedge S_1 \wedge I$
$K_4: O_1 \wedge O_2 \wedge I_4 \wedge I_5 \wedge I$

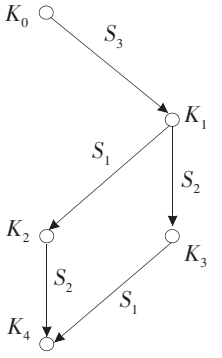


Fig.4 The automatic reasoning procedure with the restricted tableau algorithm

图 4 使用受限 tableau 算法的自动推理过程

formed into the negation normal form (NNF) and are denoted by S_1, S_2 and S_3. Finally, the automatic reasoning procedure is carried out based on the conjunction of I and S_1, S_2 and S_3 by this restricted tableau algorithm. The procedure exhaustively looks at all the possibilities of composing P_1, P_2 and P_3. If there is a model that can satisfy the conjunction and the last formula obtained from the reasoning procedure implicates the output that the required service desires, this model can be processed to obtain composite Web services meeting client requirements.

This restricted tableau algorithm is similar to the general tableau algorithm[13]. The restricted tableau algorithm is introduced to simplify the reasoning procedure. The result of the automatic reasoning procedure with this restricted tableau algorithm is depicted in Fig.4. The result of the automatic reasoning procedure with the general tableau algorithm is depicted in Fig.5. Obviously, the former is a special case of the latter and the former is simpler than the latter. The main difference between them is which formula is decomposed. For the general tableau algorithm, the decomposed formula is selected random, while for the restricted tableau algorithm, the decomposed formula is selected by the condition that the decomposed formula must represent such an atomic process whose input can be

$P_1=I_1 \wedge I_4 \sqsupseteq O_1$
$P_2=I_2 \wedge I_5 \sqsupseteq O_2$
$P_3=I_2 \wedge I_3 \sqsupseteq O_{I_4} \wedge O_{I_5}$
$(O_{I_4}=I_4, O_{I_5}=I_5)$

$I=I_1 \wedge I_2 \wedge I_3$
$S_1=\neg I_1 \vee \neg I_4 \vee O_1$
$S_2=\neg I_2 \vee \neg I_5 \vee O_2$
$S_3=\neg I_2 \vee \neg I_3 \vee (I_4 \wedge I_5)$

$K_0: S_1 \wedge S_2 \wedge S_3 \wedge I$
$K_1: \neg I_4 \wedge S_2 \wedge S_3 \wedge I$
$\quad O_1 \wedge S_2 \wedge S_3 \wedge I$
$K_2: \neg I_5 \wedge S_1 \wedge S_3 \wedge I$
$\quad O_2 \wedge S_1 \wedge S_3 \wedge I$
$K_3: I_4 \wedge I_5 \wedge S_1 \wedge S_2 \wedge I$
$K_4: \neg I_5 \wedge \neg I_4 \wedge S_3 \wedge I$
$\quad O_2 \wedge \neg I_4 \wedge S_3 \wedge I$
$\quad O_1 \wedge \neg I_5 \wedge S_3 \wedge I$
$\quad O_1 \wedge O_2 \wedge S_3 \wedge I$
$K_5: O_1 \wedge I_4 \wedge I_5 \wedge S_2 \wedge I$
$K_6: O_2 \wedge I_4 \wedge I_5 \wedge S_1 \wedge I$
$K_7: O_1 \wedge O_2 \wedge I_4 \wedge I_5 \wedge I$

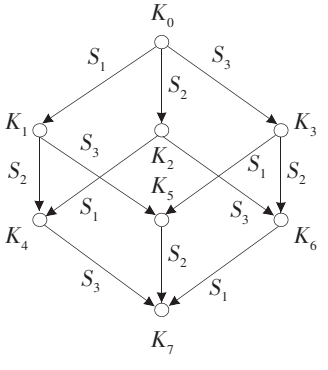


Fig.5 The automatic reasoning procedure with the general tableau algorithm

图 5 使用普通 tableau 算法的自动推理过程

represented with the true propositions contained in the formulas.

According to the following facts, we can see that the result of the restricted tableau algorithm is consistent with the results of the general tableau algorithm.

(1) Both algorithms process the same conjunction;

(2) Both algorithms try to decompose all formulas in the conjunction until no formula can be decomposed;

(3) For the general tableau algorithm, it randomly selects formulas to be decomposed, therefore, there must exist an order of decomposed formulas that is the same as the order of decomposed formulas which the restricted tableau algorithm determines.

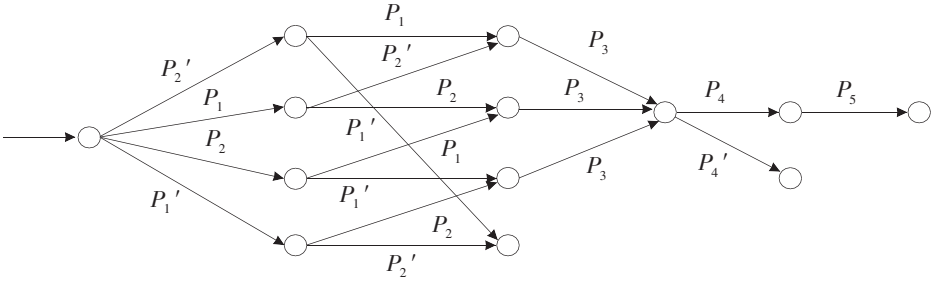
In Fig.4, each node of the graph is labeled with a set of formulas (the set may has only one formula); each arc of the graph is labeled with a formula which is decomposed. The reasoning procedure terminates until all formulas labeling the nodes of the directed graph are false or unsatisfiable or indecomposable. In this paper, "indecomposable" means that each item of the current conjunction is an atomic proposition. For the above simpler example, the reasoning procedure terminates and each item of the last conjunction (K_4) is an atomic proposition ($O_1, O_2, I_4, I_5, I_1, I_2, I_3$). For the restricted tableau algorithm, the decomposed formula is selected by the condition that the decomposed formula must represent such an atomic process whose input is implicated in the current conjunction. However, if there is no such a formula and not all items of the current conjunction are atomic propositions, the current conjunction is called unsatisfiable. In this case, the unsuitable atomic process can be found. We can delete the atomic process and try to compose those atomic processes again. Fig.6 illustrates the automatic reasoning procedure of the motivation example.

5 Modeling Web Services with SaPNs

We transform the OWL-S process models into SaPNs to represent participating services, and then give a method to represent composite Web services with SaPNs.

5.1 Representing Participating Services with SaPNs

The OWL-S process model is composed of atomic and composite processes. An atomic process represents an interaction (i.e., an indecomposable message exchange[14]) between the service and the client. Fig.3 gives the representations of atomic processes. Composite processes are composed hierarchically of some sub-processes with



P_1: $Bookname \sqsubseteq Bookbuyinfo$ P_1': $Bookname \sqsubseteq Boutstock$

P_2: $CDname \sqsubseteq CDbuyinfo$ P_2': $CDname \sqsubseteq Coutstock$

P_3: $Goodbuyinfo \wedge Creditcard \sqsubseteq \tilde{O}(\tilde{O} \sqsubseteq \tilde{I})$ P_4: $\tilde{I} \sqsubseteq Authorization \wedge Payinfo$

P_4': $\tilde{I} \sqsubseteq Forbiddance$ P_5: $Payinfo \wedge Cargotype \sqsubseteq Order$

Fig.6 The automatic reasoning procedure with the general tableau algorithm in the example

图 6 实例中使用普通 tableau 算法的自动推理过程

the control constructs, which represent a series of interactions between the service and the client. The components of a composite process are atomic processes and/or other composite processes. The OWL-S process model provides nine control constructs: *sequence*, *split*, *split+join*, *anyorder*, *if-then-else*, *choice*, *iteration*, *repeat-while* and *repeat-until*. As the iteration is not detailed enough to be instantiated in a process model, it does not be considered.

sequence: a list of components to be executed in order;

split: a set of components to be executed concurrently;

split+join: a set of components executed concurrently are synchronized;

anyorder: a set of components are executed in an unspecified order (decided at the runtime) but not concurrently;

choice: select one component to be executed from a set of components and any of the given components may be selected;

if-then-else: for two components, one is selected to be executed if a condition is true, otherwise the other is selected to be executed;

repeat-while and *repeat-until*: the former means that a component is executed when a condition is true; and the latter means that a component is executed until a condition becomes true.

The following figures are used to illustrate how atomic processes represented by SaPNs are connected with those control constructs to obtain a new SaPN to represent composite processes. To illustrate clearly, the labels of places, transitions and messages are omitted.

Fig.7 shows that two atomic processes p_1 and p_2 are executed sequentially.

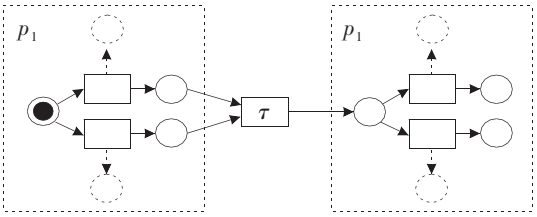


Fig.7 The atomic processes are connected with *sequence*

图 7 原子过程顺序连接

Fig.8 depicts that one of atomic processes p_1 and p_2 is selected to be executed. If the initial place represents a definite condition, p_1 and p_2 are connected with *if-then-else*, otherwise p_1 and p_2 are connected with *choice*.

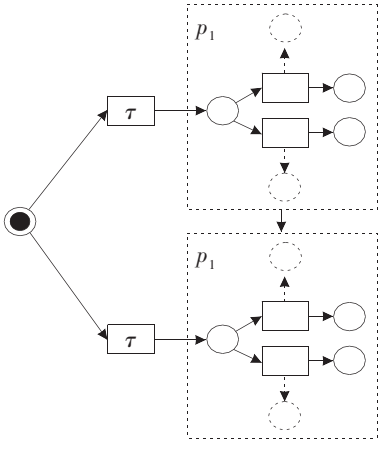


Fig.8 The atomic processes are connected with *if-then-else* and *choice*

图 8 原子过程的选择连接

The left of Fig.9 depicts that the atomic process p_1 are executed repeatedly until a condition is satisfied, while the right of Fig.9 depicts that while a condition is satisfied, the atomic process p_1 are executed repeatedly.

The left of Fig.10 depicts that two atomic processes p_1 and p_2 are executed currently without synchronization, while the left of Fig.10 depicts that two atomic processes p_1 and p_2 are executed currently with synchronization.

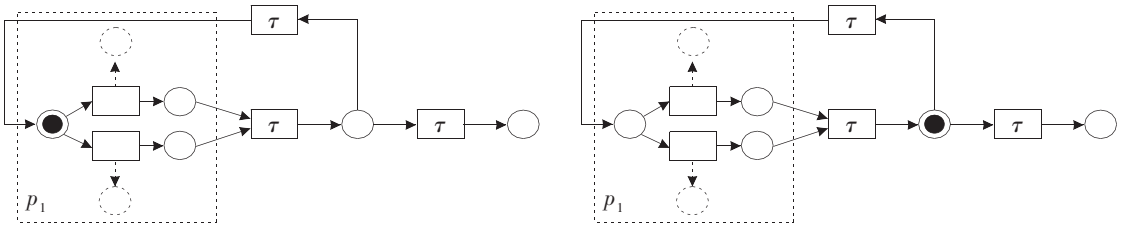


Fig.9 The atomic processes are connected with *repeat-until* and *repeat-while*

图 9 原子过程的循环连接

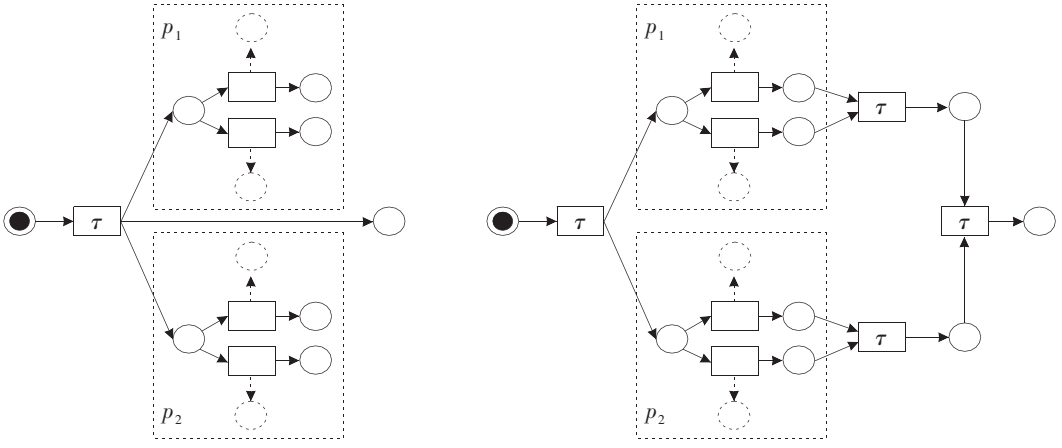


Fig.10 The atomic processes are connected with *split* and *split+join*

图 10 原子过程的并发连接

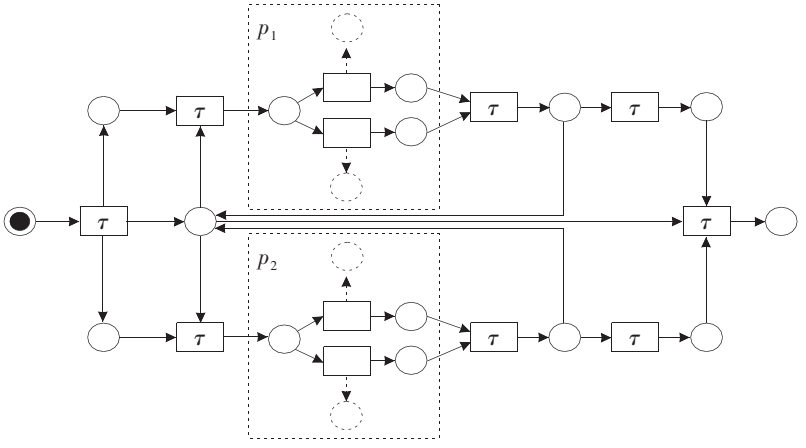


Fig.11 The atomic processes are connected with *anyorder*

图 11 原子过程的交错连接

The Fig.11 depicts two atomic processes p_1 and p_2 are executed in an unspecified order but not concurrently.

According to the above rules, the participating services *Bookseller*, *CDseller*, *Bank* and *Shippingbroker* are represented by SaPNs shown in Fig.13. Note that these SaPNs have been reduced. Since the reduction is very simple, it is not discussed here.

5.2 Representing Composite Web Services with SaPNs

As Fig.3 shows, an atomic process can be repre-

sented by a SaPN. According to the temporal sequences of atomic processes defined by the composite Web service, these SaPNs representing atomic processes are connected. As a result, a new complex SaPN can be obtained to represent the composite Web service. Fig.6 gives the temporal sequences of atomic processes that compose the required service meeting the client requirements. With this method, Fig.12 gives a complex SaPN representing this required service.

6 Verifying Composite Web Services

In this Section, the properties of SaPNs are employed to verify composite Web services obtained in Section 5. That is to say, we will check whether deadlock, unsafeness or non-reachability occurs in the interactions between the required service and participating services *Bookseller*, *CDseller*, *Bank* and *Shippingbroker*, firstly, a SaPN is employed to represent the interactions.

Definition 5 Let $SN_i=(P_i,T_i,F_i,\Sigma_i,M_{i0},a_i)$ $(1\leqslant i\leqslant n)$ be SaPNs representing all participating services of a composite Web service and this composite Web service is represented by a SaPN $SN=(P,T,F,\Sigma,M_0,a)$. Thus, we can obtain a new SaPN $SN_0=(P',T',F',\Sigma',M_0',a')$

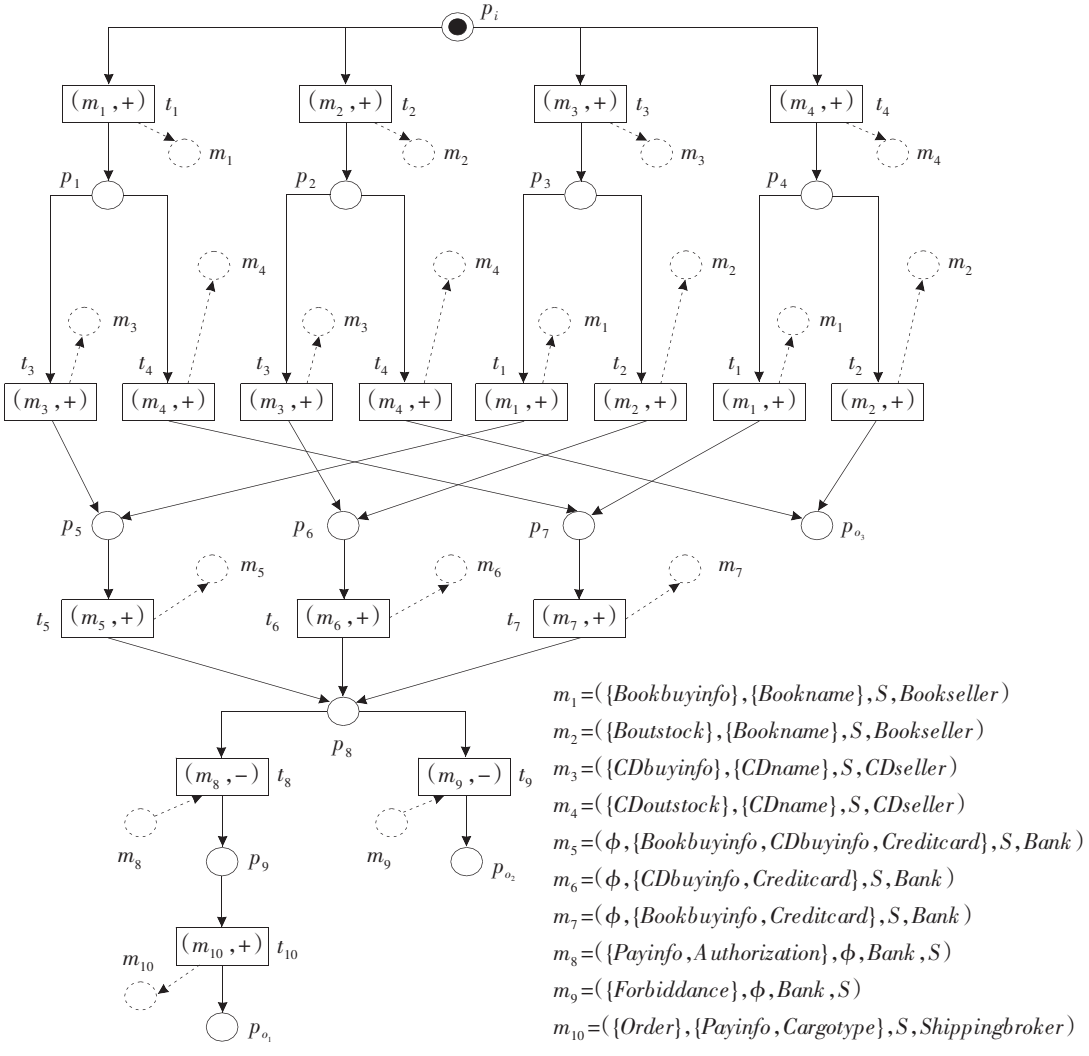


Fig.12 The composite Web service represented by a SaPN

图 12 使用 SaPN 描述组合 Web 服务

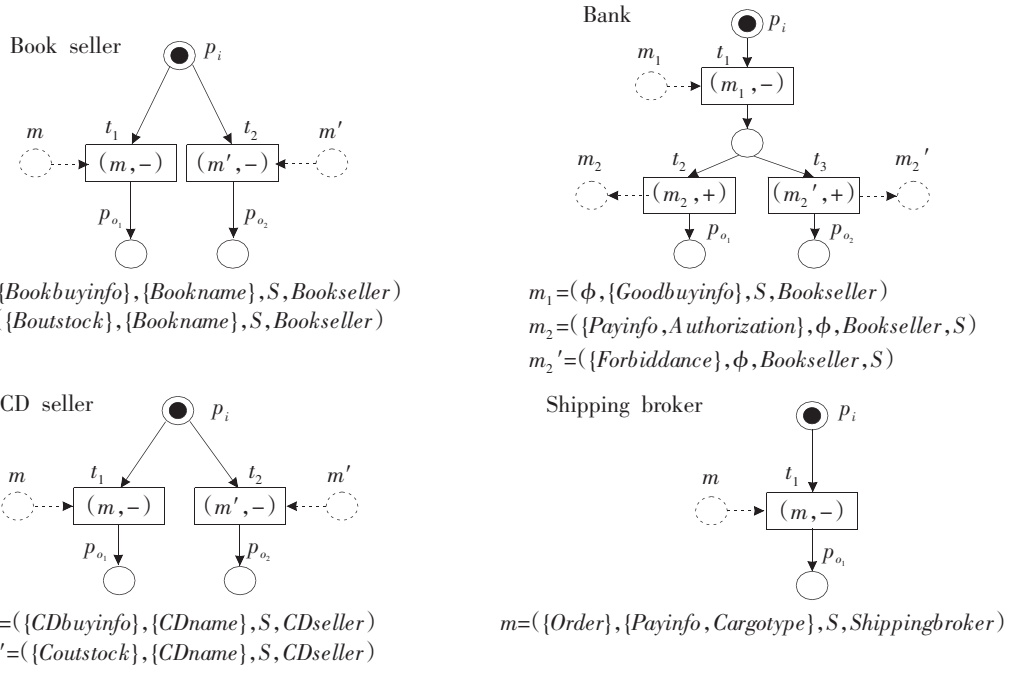


Fig.13 The participating services represented by SaPNs

图 13 使用 SaPN 描述组成部分

representing the interactions among these services, where:

$$P'=\bigcup_{1\leqslant i\leqslant n}P_i,\ T'=\bigcup_{1\leqslant i\leqslant n}T_i,\ F'=\bigcup_{1\leqslant i\leqslant n}F_i;$$

$\Sigma'=\bigcup_{1\leqslant i\leqslant n}\Sigma_i$, for any two messages (I_i,O_i,S_i,R_i) and (I_j,O_j,S_j,R_j) $(i\neq j)$ in Σ'; if $Match(I_i,I_j)\geqslant threshold$ and $Match(O_j,O_i)\geqslant threshold$[1] furthermore, $S_i=S_j$ and $R_i=R_j$, then $(I_i,O_i,S_i,R_i)=(I_j,O_j,S_j,R_j)$;

$M_0'=(M_{c0}',M_{d0}')$, where $M_{c0}'=(M_{1c0},M_{2c0},\cdots,M_{nc0})$, $M_{d0}'=M_{1d}'=M_{2d}'=\cdots=M_{nd}''$;

$\forall t\in T,\ a(t)=a_i(t)$, if $t\in T_i$.

In the following, Fig.14, 15 and 16 illustrate deadlock, unsafeness and non-reachablility problem respectively occurring in the interactions among the services. Fig.14 shows that the interactions between the services S_1 and S_2 cannot terminate normally, because the data conditions for t_1 and t_2 never can be satisfied. Fig.15 illustrates that the message m_1 makes no sense to the interactions (m_1 cannot be consumed by other services) so that the channel stored m_1 might overflow. Fig.16 depicts that the end place p_{o_1} never can be reached in terms of the interactions between S_1 and S_2, since no service can produce the message m. Thus, some expected results cannot be obtained.

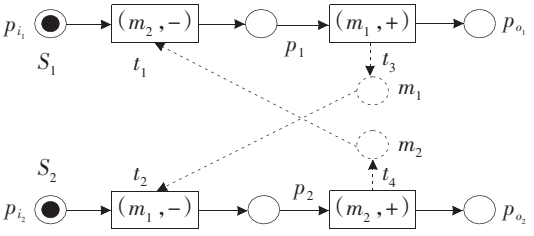


Fig.14 The deadlock problem occurs in the interactions between two services

图 14 发生在两个服务间交互的死锁问题

[1] $Match(A,B)\geqslant threshold$ means that $\forall a\in A$, there exists $b\in B$ such that $a\sqsupseteq b$, where a and b are concepts in an ontology[12].

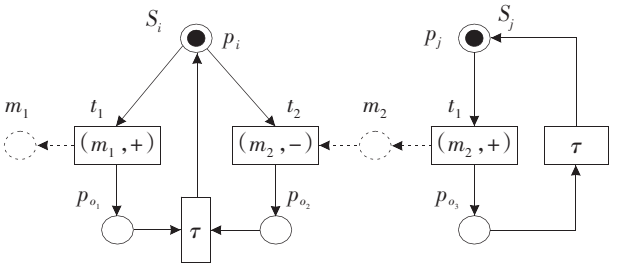


Fig.15 The unsafeness problem occurs in the interactions between two services

图 15 发生在两个服务间交互的不安全问题

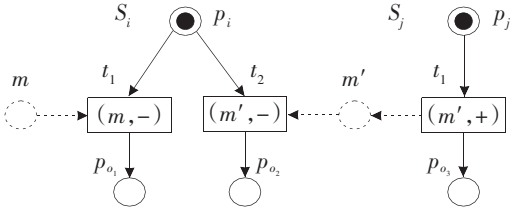


Fig.16 The non-reachability problem occurs in the interactions between two services

图 16 发生在两个服务间交互的不可达问题

Definition 6 Given a semantic annotated Petri Net (P,T,F,Σ,M_0,a) representing the interactions among services, a marking M_n is called reachable from M_0 (notation $M_0 \overset{*}{\rightarrow} M_n$) iff there is a firing sequence $\sigma = t_1 t_2 t_3 \cdots t_n$ such that $M_0 \overset{t_1}{\rightarrow} M_1 \overset{t_2}{\rightarrow} M_2 \cdots \overset{t_n}{\rightarrow} M_n$.

Note that the empty firing sequence is also allowed, i.e., $M_1 \overset{*}{\rightarrow} M_1$.

Definition 7 A semantic annotated Petri Net (P,T,F,Σ,M_0,a) is live iff, for every reachable marking M and every transition t, there is a marking M' reachable from M which enables t.

A SaPN is structurally live if there is an initial marking such that the net is live.

Definition 8 A SaPN (P,T,F,Σ,M_0,a) is bounded iff, for each place p there is a natural number n such that for every reachable marking the number of tokens in p is less than n. The net is safe iff for each place the maximum number of tokens does not exceed 1.

A SaPN is structurally bounded if the net is bounded for any initial marking. Since the technologies for checking whether deadlock, unsafeness and non-reachability occurs in Petri Nets is well known[15], it does not be discussed here.

7 Conclusion

In this paper, an approach for modeling and verifying composite Web services based on semantic annotated Petri Nets is proposed. Firstly, semantic annotated Petri Nets (SaPNs) and the semantics are presented. Secondly, a restricted description logic tableau algorithm is utilized to obtain composite Web services. Thirdly, SaPNs is used to represent a composite Web service and the participating services that are invoked by this composite Web service. Finally, the composite Web service is verified by means of some analysis methods based on SaPNs. With this approach, a reliable composite Web services meeting client requirements in an open Internet environment can be achieved. In the future, some tools which are the basis for using this approach widely and practically in service-oriented computing will be developed.

References:

[1] Booth D, Haas H, McCabe F, et al. Web services architecture[EB/OL]. [2007-09-10]. http://www.w3.org/TR/ws-arch/.

[2] Chinnic R, Gudgin M, Moreau J, et al. Web services description language version 1.2 [EB/OL]. [2007-09-10]. http://www.w3.org/TR/2003/WD-wsdl2-20030124/intro.

[3] Mitra N, Lafon Y. Soap version 1.2 part 0: Primer[EB/OL]. (2007-04-27) [2007-09-10]. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

[4] Ariba I, Corporation I, Corporation M. Universal description, discovery and integration[EB/OL]. [2007-09-10]. http://www.uddi.org/.

[5] Andrews T. Business process execution language for Web services version 1.1[EB/OL]. [2007-09-10]. http://www.ibm.com/developeworks/library/ws-bpel/.

[6] Lohmann N, Kopp O, Leymann F, et al. Analyzing BPEL4Chor: Verification and participant synthesis [C]//Dumas M, Heckel R. LNCS 4937: Fourth International Workshop on Web Services and Formal Methods: Proceedings of WS-FM 2007. Berlin: Springer-Verlag, 2008:46-60.

[7] Huang Yu, Wang Hanpin. A Petri Net semantics for Web service choreography[C]//Chbeir R, Claramunt C, Li K, et al. Symposium on Applied Computing: Proceedings of the 2007 ACM Symposium on Applied Computing. New York: ACM, 2007:1689-1690.

[8] Kang Hui, Yang Xiuli, Yuan Sinmiao. Modeling and Verification of Web services composition based on CPN [C]//Li Keqiu, Xiang Yang. Proceedings of International Conference on Network and Parallel Computing Workshops 2007. Los Alamitos: IEEE Computer Society, 2007:613-617.

[9] Hiddersa J, Kwasnikowskac N, Srokab J, et al. DFL: A data-flow language based on Petri Nets and nested relational calculus[J]. Information Systems, 2008,33(3):261-284.

[10] Martin D, Burstein M, Hobbs J, et al. OWL-S: Semantic markup for Web services[EB/OL]. (2004-11-22) [2007-09-10]. http://www.w3.org/Submission/OWL-S/.

[11] Lei Lihui, Duan Zhenhua. Semantic matching of Web services for collaborative business processes[C]//Shen Weiming, Luo Junzhou, Lin Zongkai, et al. LNCS 4402: Proceedings of the 2006 10th International Conference on Computer Supported Cooperative Work in Design. Berlin: Springer-Verlag, 2007: 479-488.

[12] Paolucci M, Kawamura T, Payne T, et al. Semantic matching of Web services capabilities[C]//Horrocks I, Hendler J. LNCS 2342: Proceedings of the ISWC 2002. Berlin: Springer-Verlag, 2002:333-347.

[13] Baader F, Calvanese D, McGuinness D, et al. The description logic handbook: Theory, implementation and applications[M]. Cambridge: Cambridge Press, 2003.

[14] Lei Lihui, Duan Zhenhua. Transforming OWL-S process model into EDFA for service discovery[C]//Leymann F, Zhang Liangjie, Feig E, et al. Proceedings of IEEE International Conference on Web Services 2006. Washington: IEEE Computer Society, 2006:137-144.

[15] Aalst W. Workflow verification: Finding control-flow errors using Petri-Net-based techniques[C]//Goos G, Hartmanis J, Leeuwen J. LNCS 1806: Business Process Management: Models, Techniques, and Empirical Studies. Berlin: Springer-Verlag, 2000:161-183.

LEI Lihui was born in 1976. She received the Ph.D. degree in Computer Science from Xidian University in 2008. Her research interests include Internet computing, semantic Web, etc.

雷丽晖(1976-),女,陕西西安人,博士,主要研究领域为网络计算,语义 Web。

DUAN Zhenhua was born in 1948. He received two Ph.D. degrees in Computer Science from Newcastle University in 1996 and Sheffield University in 1997. He is a professor and doctoral supervisor at Xidian University. His research interests include Internet computing, theory and technology of software reliability, etc.

段振华(1948-),男,博士,教授,博士生导师,主要研究领域为网络计算,可信软件理论和技术。在国内外会议、期刊上发表论文 100 多篇,承担过国家自然科学基金重点和面上项目等研究课题。