Petri nets-based Models for Web Services Composition*

Huaikou Miao[1,2] and Tao He[1]
[1]School of Computer Engineering and Science, Shanghai University
[2] Shanghai Key Laboratory of Computer Software Testing and Evaluating
[1,2] P. R. China

1. Introduction

Web services receive significant research recently from both academia and industry due to its broad applications and flexible architecture supporting recomposition and reconfiguration lately. Research on semantic web brings new energy to Web services; combining semantic web with Web service will be a kind of intelligent Web service which is the trend of Web service development. An individual service, whose functionality is limited, cannot meet practical application's needs, so composing Web services and generating a new value-added Web service, more functionalities can be provided and the potential of Web services can be showed. Composing Web services is to find available Web services, integrate interfaces between Web services, combine several autonomous Web services together in term of requirements of applications, and provide a more powerful composite service.

In order to allow flexible automation and composition of semantic representations of Web services, OWL-S (OWL for Services) was proposed. OWL-S service descriptions (e.g., the service process model) provide the needed information for a priori analysis and verification of service invocations and compositions. Due to the lack of formal semantics in the OWL-S specification, McIllraith and Narayanan use Petri nets to test and verify the composition of Web services based on OWL-S (Martin D.etal.2004). Some researches are currently being done on automated provision and reasoning about Web services. (Cordoso,J &Sheth, A. 2002)

Rachid Hamadi (Rachid Hamadi, Boualem Benatallah, 2003) proposed a Petri nets-based algebra for composing Web services; any service that is expressed using the algebra constructs, can be translated into a Petri nets model. This model without semantics belongs to the composition based on service level, particle is too large. Zhang Jia (Zhang Jia et al.,2004) presented a WS-net model based on colored Petri nets, described Web service components from three aspects which are the interface net, the interconnect net and the

* This work was supported in part by a grant from National Natural Science Foundation of China(NSFC) under grant No. 60673115; National High-Technology Research and Development Program(863 Program) of China under grant No. 2007AA01Z144; National Grand Basic Research Program(973 Program) of China under grant No. 2007CB310800; Research Program of Shanghai Education Committee under grant No. 07ZZ06; Shanghai Leading Academic Discipline Project, Project Number: J50103.

interoperation net. This model that supports the object-oriented paradigm and the component-based concept, is convenient to verify and stimulate service composition, but preconditions and effects of services are not considered.
Narayanan et al. (S.Narayanan &S.Mcllraith, 2002) defined semantics of DAML-S atomic processes in terms of a set of situation calculus axioms, and every basic service was represented by a situation calculus formula whose operational semantics was provided using Petri nets. Moreover, verification of the composite service can be realized by reachability of Petri nets. Nevertheless, it did not discuss about composite planning and process model. S. Narayanan and S. A. Mcllraith proposed a model based on Petri nets to specify a Web service of OWL-S (J.P. Thomas et al., 2003). They supposed that the service was composed of several atomic services, and the supposition was not suitable in open-environment composite service. Furthermore, the model was not described clearly in their proposition.
This chapter mainly discusses how to model and analyze the composite processes of semantic Web services using Petri nets. In this model, input, output and precondition are represented through different kinds of tokens; effect is represented by the change of the token number during firing the transition. The composition of two or more services generates a new service providing both the original individual behavioral logic and a new collaborative behavior for carrying out a new composite task. A composite service consists of a collection of Web services related by data and control flow. Composition models can be described unambiguously and composite processes can be analyzed and verified conveniently.

2. Background

2.1 OWL-S

OWL-S is defined as a W3C standard to provide a computer-interpretable description of the services, service access and service composition using OWL ontologies. It is an upper ontology for modelling web-service composition which offers a process-based perspective. OWL-S provides declarative publications of services properties and capabilities, API for Web services, specifications of prerequisites and consequences of individual services and descriptors for the state of services execution for automatic services discovery, invocation and execution monitoring.

OWL-S process model describes formation of services by composition

(i) Service profile –which presents what function the service computes. This information is expressed in terms of the transformation that the service produces.

(ii) Process model – which describes the service behavior providing a view of the service in terms of process compositions. OWL–S defines three types of processes:

atomic processes, which have associated inputs and outputs and can be directly invoked by the client,

composite processes, which consist of other composite and atomic processes, and

simple processes, which are abstract and simplified view of a composite Process.

(iii)Service grounding – which offers all details about their invocation.

An atomic process cannot be decomposed further and it executes in a single step (similarly to a black box providing a functionality), while a composite process is built up by using a

few control constructs: Sequence, Split, Split-Join, Any-Order, Iterate, If-Then-Else, Choice, repeat-while and repeat-until. Hence, for instance, an if-then-else process is a bag of two processes out of which one is chosen for execution according to the value of a condition, an any-order process is a bag of processes to be executed in some unspecified order but not concurrently, and a repeat-until process is a process to be executed at least one, until a condition becomes true.

2.2 Petri nets concept

Petri nets model has a strong capability to model events and states in a distributed system and to capture sequential, concurrency and event-based control. Petri net is also a powerful tool for analyzing and verifying certain properties such as reachability, liveness, and deadlocks (Maurice ter Beek et al., 2007). For examples, the Marking Reachability relation identifies the reachability between any two markings; the construct inclusion relation identifies the inclusions between two OWL-S construct by analyzing their Petri nets representations.

The Petri nets model is a bipartite graph containing places representing states and transitions representing actions. Places hold tokens that represent predicates of the state. A transition will be trigger when all the places pointing to the transition obtain an adequate number of tokens.

A place/transition Petri nets is represented as a quintuple PN = (P, T, F, W, M_0), where:

$P=\{p_1,p_2,\ldots,p_m\}$ is a finite set of places.

$T=\{t_1,t_2,\ldots,t_m\}$ is a finite set of transitions that represents the set for tasks, processes, activities, or events.

F is a set of arcs used to represents flow paths, where $F\subseteq(P\times T)\cup(T\times P)$

$W : F\rightarrow (\mathbb{N} \setminus \{0\})$ is the arc weight mapping. $\mathbb{N}$ is natural numbers set.

M_0: $P\rightarrow\mathbb{N}$ is an initial marking, where every place has a number of tokens.

Given a marking M, a transition t is *enabled* in M If and only if $M(p)\neq 0$ for each $p\in\bullet t$. t is fired in M If and only if it is enabled in M and M is transformed into M' such that (i)$\forall p\in \bullet t: M'(p)=M(p)-1$, (ii) $\forall p\in t\bullet: M'(p)=M(p)+1$, and (iii) $\forall p \notin \bullet t\bullet$: $M'(p)= M(p)$. In this case, M' is *directly reachable* from M via t, denoted as M[t>M'. M' is directly reachable from M, denoted as M[>M', If and only if M[t>M' for some $t\in T$. Given $\sigma\in T^*$, M' is *reachable* from M viaσ, denoted as M[σ>M', If and only if (i) M'=M when $|\sigma|=0$, or (ii) $\sigma=t_1t_2\ldots t_k$, k>0 and there exists a sequence $M_0[t_1>M_1[t_2>\ldots M_{k-1}[t_k>M_k$ such that M_0=M and M_k=M'. In this case, $\sigma$is called a firing sequence from M to M'. M' is said to be a reachable marking in (N,M_0), and$\sigma$is called a firing sequence of M. The set of trackable markings in (N,M_0) is denoted as $RM(N,M_0)$. The corresponding reachability graph is denoted as $RG(N,M_0)$.

Given a Petri nets PN=(N,M_0), PN is *bounded* If and only if $RG(N,M_0)$ is finite, i.e., $\exists k\geq 0$ such that $\forall M\in RG(N,M_0)$ $\forall p\in P: M(p)\leq K$. In this case, we also say PN is K-bounded. PN is safe If and only if it is 1-bounded. PN is *live* (or M_0 is a live marking) If and only if $\forall M\in RG(N,M_0)$ $\forall t\in T$, $\exists M'\in RM(N,M_0): M[>*M'$ and t is enabled in M′. A reachable marking M is a deadlock marking If and only if there are no transitions are enabled in M.

To represent an OWL–S process model with Petri nets, we consider atomic processes as transitions and an atomic process can be executed only if the following two conditions occur:

(i) all of its inputs are available, and

(ii) all processes to be executed before it have been completed.

3. P/T Petri nets-based models for Semantic Web services composition

3.1 From OWL-S to Petri nets

The Web Service application logic described by OWL-S is first transformed into a Petri nets model to provide a formal representation of the structure and behavior of the service. Petri nets ontologies are defined to carry the operational semantic as well as the inputs, outputs, precondition and effects (IOPE) semantics of OWL-S service functionality.

The processes organized by OWL-S control constructs are mapped to the Petri nets by analyzing their execution semantics, the Perform actions, and the IOPE of each Perform. Its inputs are mapped to the places holding tokens pointing to the transition. Its preconditions are mapped to corresponding arc labels that must be valid in order to enable the transition. The Perform effects and output are mapped to output arcs and places of the transition that will be triggered after the occurrence of the Perform transition.

OWL-S distinguishes between atomic and composite processes. Atomic processes are indivisible processes that result in a message exchange between the client and the server. Composite processes are used to describe the control flow relation between processes.

The more general the model, the less amenable it is to analysis, so we extend classic Petri nets to be Web Service Petri nets for Web services verifications. The followings are definitions:

Definition 1 The Petri nets model for an atomic process described by OWL-s is a Petri nets $BN = (S, T; F, W, M_0)$ as shown in Fig. 1, where

$S=\{s\}$. It represents the service to be run when *s* includes tokens.

$T=\{t_b,t_e\}$, where, tb represents beginning of service and t_e represents accomplishment of service, corresponding to Precondition and Effects respectively,

$F=\{(t_b,s), (s, t_e)\}$,

$M_0=0$,

$W(t_b,s)$ and $W(s, t_e)$ represent Input and Output respectively,

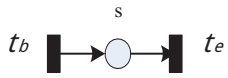


Fig. 1. Atomic process model

Definition 2 The Petri nets model for sequence composition of Web services is a hierarchical Petri nets $CBN= (S, BN; F, W, M_0)$ as shown in Fig. 2, where

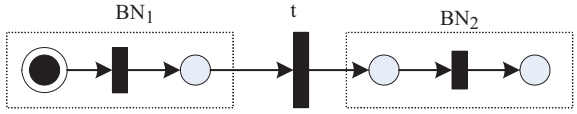


Fig. 2. Sequence model

$S = \{s_1, s_2, \ldots, s_k\}$,

$BN=\{BN_1, BN_2, \ldots, BN_k\}$,where BN_i is a subnet of the i-th service.

$F=\{(BN_i, t_i), (t_i, BN_{i,i+1})\}$,

$M_0=\{M_0(s_i) \mid i=1,2,\ldots,k-1\}=0$,

$W=\{W(BN_i, s_i), W(s_i, BN_{i+1}) \mid i=1,2,\ldots,k-1\}$,

Definition 3 The Petri nets model for split composition of Web services is a hierarchical Petri nets $BN= (S, BN; F, W, M_0)$ as shown in Fig. 3, where

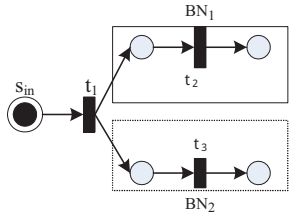


Fig.3. Split model

$S = \{s_{in}\}$,

$T=\{t_1\}$

$BN=\{BN_1, BN_2, \ldots, BN_k\}$,where BNi is a subnet of the i-th service.

$F=(s_{in}, t_1) \cup (t_1, BN_i)$,

$W= W(s_{in}, t_1) \cup W(t_1, BN_i)$, $i=1,2,\ldots,k$,

$M[t_1>M_1 \rightarrow M_1[t_2>$,and $M[t_1>M_2 \rightarrow M_2[t_3>$.

Definition 4 The Petri nets model for Split+Join composition of Web services is a hierarchical Petri nets $CBN= (S, BN; F, W, M_0)$ as shown in Fig. 4, where,

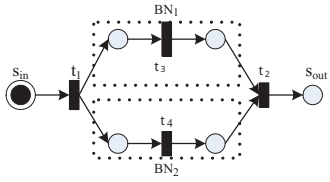


Fig. 4. Split-join model

$S = \{s_{in}, s_{out}\}$,

$T=\{t_1, t_2, t_3, t_4\}$

$BN=\{BN_1, BN_2, \ldots, BN_k\}$,where BN_i is a subnet of the i-th service.

$F=(s_{in}, t_1) \cup (t_1, BN_i) \cup (BN_i, t_2) \cup (t_1, s_{out})$,

$W= W(s_{in}, t_1) \cup W(t_1, BN_i) \cup W(BN_i, t_2) \cup W(t_1, s_{out})$, $i=1,2,\ldots,k$,

$M[t_1>M_1 \rightarrow M_1[t_2>$,and $M[t_1>M_2 \rightarrow M_2[t_3>$

Then $M_1[t_3>$ and $M_2[t_4> \rightarrow M_3[t_4>$.

Definition 5 The Petri nets model for Choice composition of Web services is a hierarchical Petri nets $CBN= (S, BN; F, W, M_0)$ as shown in Fig. 5, where

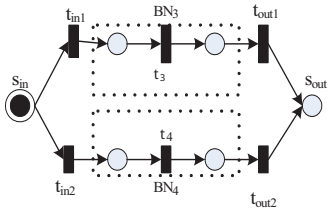


Fig. 5. Choice model

$S = \{s_{in}, s_{out}\}$,

$T=\{t_{ini}, t_{outi}\}$

$BN=\{BN_1, BN_2, \ldots, BN_k\}$, where BN_i is a subnet of the i-th service.

$F=(s_{in}, t_{ini}) \cup (t_{ini}, BN_i) \cup (BN_i, t_{outi}) \cup (t_{outi}, s_{out})$,

$W= W(s_{in}, t_{ini}) \cup W(t_{ini}, BN_i) \cup W(BN_i, t_{outi}) \cup W(t_{outi}, s_{out})$, i=1,2,…,k,

$M[t_{in1}> M_1 \rightarrow \lceil M_1\ [t_2>$ or $M[t_{in2}> M_2 \rightarrow \lceil M_2[t_{in1}>$.

Definition 6 The Petri nets model for If-Then-Else composition of k Web services is a hierarchical Petri nets BN= (S, BN; F, W, M_0) as shown in Fig. 6, where

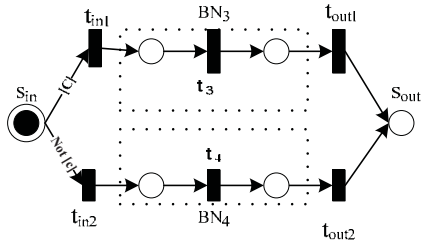


Fig. 6. If-then-else model

$S = \{s_{in}, s_{out}\}$,

$T=\{t_{ini}, t_{outi}\}$

$CBN=\{BN_1, BN_2, \ldots, BN_k\}$, where BN_i is a subnet of the i-th service.

$F=(s_{in}, t_{ini}) \cup (t_{ini}, BN_i) \cup (BN_i, t_{outi}) \cup (t_{outi}, s_{out})$,

$W= W(s_{in}, t_{ini}) \cup W(t_{ini}, BN_i) \cup W(BN_i, t_{outi}) \cup W(t_{outi}, s_{out})$, $W(s_{in}, t_{in1})=c$, $W(s_{in}, t_{in2})\neq c$, i=1,2,…,k,

$M[t_{in1}> \rightarrow \lceil M[t_2>$ or $M[t_{in2}> \rightarrow \lceil M[t_{in1}>$

Definition 7 The Petri nets model for Repeat-While composition of k Web services is a hierarchical Petri nets CBN= (S, BN; F, W, M_0) as shown in Fig. 7, where

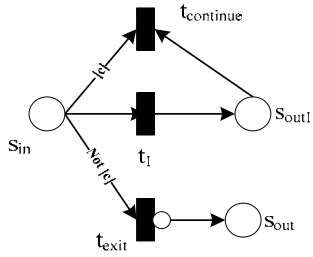


Fig. 7. Repeat-while model

$S = \{s_{in}, s_{out1}, s_{out}\}$,

$T=\{t_1, t_{continue}, t_{exit}\}$

$CBN=\{BN_1, BN_2, \ldots, BN_k\}$,where BN_i is a subnet of the i-th service.

$F=(s_{in},t_1) \cup (t_1, s_{out1}) \cup (s_{out1}, t_{continue}) \cup (t_{continue}, s_{in}) \cup (s_{in},t_{exit}) \cup (t_{exit}, s_{out})$,

$W= W(s_{in},t_1) \cup W(t_1, s_{out1}) \cup W(s_{out1}, t_{continue}) \cup W(t_{continue}, s_{in}) \cup W(s_{in},t_{exit}) \cup W(t_{exit}, s_{out})$, $W(s_{in},t_1)=c$, $W(s_{in},t_{exit})\neq c$, $i=1,2,\ldots,k$,

$M[t_1> \rightarrow \urcorner M[t_{exit}>$ or $M[t_{exit}> \rightarrow \urcorner M[t_1>$

(t_{exit}, s_{out}) is an inhibitor arc. An inhibitor arc connects a place to a transition and is represented by a line with a small circle instead of an arrowhead at the transition. The inhibitor arc disables the transition when the input place has a token and enables the transition when the input place has no token and other input places have at least one token per arc weight. No tokens are moved through an inhibitor arc when the transition fires.

Definition 8 The Petri nets model for Repeat-Until composition of k Web services is a hierarchical Petri nets $CBN= (S, BN; F, W, M_0)$ as shown in Fig. 8, where

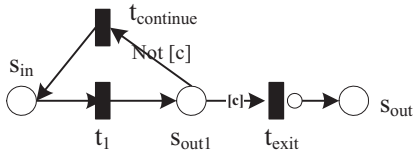


Fig. 8. Repeat-until mode**l**

$S = \{s_{in}, s_{out1}, s_{out}\}$,

$T=\{t_1, t_{continue}, t_{exit}\}$

$BN=\{BN_1, BN_2, \ldots, BN_k\}$,where BN_i is a subnet of the i-th service.

$F=(s_{in},t_1) \cup (t_1, s_{out1}) \cup (s_{out1}, t_{continue}) \cup (t_{continue}, s_{in}) \cup (s_{out1},t_{exit}) \cup (t_{exit}, s_{out})$,

$W= W(s_{in},t_1) \cup W(t_1, s_{out1}) \cup W(s_{out1}, t_{continue}) \cup W(t_{continue}, s_{in}) \cup W(s_{out1},t_{exit}) \cup W(t_{exit}, s_{out})$, $W(s_{out1},t_{exit})=c$, $W(s_{out1}, t_{continue})\neq c$, $i=1,2,\ldots,k$,

(t_{exit}, s_{out}) is an inhibitor arc.

The Petri nets model captures the structure and operational semantics of composite WS described by OWL-S process model. OWL-S Processes can be mapped to Petri nets and generalized as follows:

Process ::= AtomicProcess ... |CompositeProcess CProcess ...

CProcess ::= AnyOrder PerformanceList |

Sequence PerformanceList |
Split PerformanceList |
SplitJoin PerformanceList |
Choice PerformanceList |
IfThenElse Performance|
RepeatWhile Peformance |
RepeatUntil Performance | Connect ...
Performance ::= Perform Process.

3.2 Composition Net

Most systems that arise from practical applications are very complex and practically unmanageable.(Zhijun Ding et al.,2005) For this reason, modular construction methods provide a mechanism to manage the complexities of a large system that can be built out of well understood smaller subsystems. One way to do this is through Petri nets synthesis based on some prescribed construction rules which preserve certain logical properties as the construction progresses. These subsystems are then combined through common places and/or transitions into a larger subsystem at each synthesis step. Each subsystem is modeled separately while ignoring interactions with other subsystems. These subsystems are then combined through common places and/or transitions into a larger subsystem at each synthesis step.

Every interface place represents either messages from the service to a partner, or messages from a partner to the interface. The service is connected to the interface place in only one direction. The interface net identifies each subsystem as a unique functional object, and the interconnection net specifies the relationships between subsystems. As a result, we can visualize the entire topological view of a system by interconnecting each of the interconnection nets according to our unique module-interconnection technique. Furthermore, we assume that a service reads or writes only one message per transition. It may, however, perform transitions that do not interact with the interface at all. The concept of module formalizes our view on Web services as workflow modules equipped with an interface.

Definition 9 (Module) MD=(P, T, F,W,m_0) is a module if

(i) P=(Place, Port), Place=(P_s,P_e,P_M), P_s is the start and P_e is the end place respectively. P_M is the set of internal places. Port=($Port_I$, $Port_O$), $Port_I$ is the set of input ports, and $Port_O$ is the set of output ports, which are pairwise disjoint.

(ii) (P, T, F,W, m_0) is a Petri nets with $m_0(p_s)$=1 and $m_0(p)$=0 for all other places p,

(iii) for all places and transitions x, (p_s,x) and (x, p_e) are in the reflexive and transitive closure of F,

(iv) every write transition is connected to exactly one output port and no input port, every read transition is connected to exactly one input port and no output port, every internal transition is connected to neither an input nor an output port.

We model the system behavior w.r.t abstraction BN by a Petri nets N=(P,T) as follows. Let J to be a sub-system, and sub-system J∈M is modeled as a subnet BN_i=(BP_i,BT_i) of N, called the blackbox Petri nets of J. Suppose J has m inputs and n outputs, the corresponding BN_i consist of five parts.

Given an abstraction subsystem BN, let BNi be the corresponding Petri nets. We conduct reachability analysis for BN based on some initial marking M_0. Denote $RG(N,M_0)$ as the resulting reachability graph. We check that the following conditions hold for $RG(N,M_0)$:

(1) $RG(N,M_0)$ is finite.

(2) $M_0(p_M)=0$ for each BN_i in N.

(3) For each reachable marking M, for each blackbox Petri nets BN_i in BN with m inputs and n outputs, the following two conditions hold: (1)$\forall i\in[1\ldots m]:M(p_i)<1$ (2) If $\exists i\in[1\ldots m]:M(p_i)=1$, then $\forall j \in [1..m], j \neq i : M(p_j)$ $\forall j\in[1\ldots m], j\neq i:M(p_j)$

Web service net provides an interconnection mechanism across different levels of component diagrams. Interconnections can be visualized by: (1) interoperation nets of sender and receiver components, and (2) the interface net of the sender, receiver, and channel components. We believe that this is a very important feature to visualize very large systems. By applying such visual abstractions, such as replacing large interoperation nets with simpler interconnection nets or even with interface nets, complicated nets can be effectively visualized at various levels of abstraction.

Theorem 1 Let $PN=(P,T;F,W,M_0)$ be a Petri nets model of Web service composition. $R(M_0)$ is the reachability set of M_0, then $R(M_0)$ is a finite set.

Proof: According to the definition of P/T Petri nets, for each reachability marking $M:M\in R(M_0)$, M(p) is a set of tokens residing in a place $p\in P$, including a tuple of symbolic individuals or structure terms constructed from individuals and operations. Moreover, for a service composition, the number of individuals and variables is always finite, so the number of tokens in M(p) is finite. At the same time, P is a finite set of places, and obviously the combination of a finite number of tokens with a finite number of places is always finite. Hence the number of reachable marking is finite, that is, $R(M_0)$ is a finite set.

Theorem 2 Let M be a marking in $RG(N,MB_{0B})$, $MB_{1B}[\sigma>MB_{2,B}$, p is an internal place of $JB_{.B}$, For each $J\in BN$, M is reachable in $RG(N,MB_{0B})$ via a canonical execution sequence.

Proof: Suppose $MB_{2\,B}$ is reachable from MB_{1B} via σ in $RG(N,MB_{0B})$, where $MB_{1B}(p)=0$. Let $k=|\sigma P^{outP}|$. Then $k\leq|\sigma P^{inP}|\leq k+1$. In addition, MB_{2B} is reachable from MB_{1B} in $RG(N,MB_{0B})$ via $\eta=\eta B_{0B}\eta B_{1B}\ldots\eta B_{kB}\eta B_{k+1B}$. When $MB_{1B}=MB_{0B}$, it is a canonical execution sequence for reachable marking MB_{2B} w.r.t BN. Since $MB_{1B}(p)=0$, by the structure of BN, there must be at least k input transitions of BN in σ, and for each $l\in[1\ldots k]$, the *l*-th input transition of BN must occur before the *l*-th output transition of BN in σ, which can be written as $\eta B_{0B}\sigma B_{1B}\ldots\sigma B_{kB}\eta B_{k+1B}$. So an execution sequence σ from MB_{1B} to MB_{2B} is canonical execution sequence. Since $MB_{0B}(p)=0$, any execution sequence for a reachable marking M can be rewritten into its canonical form w.r.t BN.

Given two abstractions BN and BN′, BN′ is called a one-step refinement of BN, denoted as $BN\prec BN'$, If and only if $BN'=(BN\setminus\{J\})\cup\{J_1,J_2,\ldots,J_k\}$,$k\geq2$, where $\{J_1,J_2,\ldots,J_k\}$ is the set of component subsystems of J via one step decomposition.

Theorem 3 Given Petri nets $N\prec N'$. Let $RG(N,M_0)$ and $RG(N',M_0')$ be the corresponding reachability graphs of N and N', respectively. The following statements are true:

- Deadlock: $RG(N,M_0)$is deadlock free If and only if $RG(N',M_0')$ is deadlock free.
- Liveness: A transition $t\in T$ is live in $RG(N,M_0)$ If and only if it is live in $RG(N',M_0')$.

Proof: Deadlock: Suppose M is a deadlock marking in RG(N, MB_{0B}). Let σ be a firing sequence for M. Then no transition in T is enabled in M. In particular, M(p)=0, and $|\sigma P^{inP}|=|\sigma P^{outP}|$. There is a marking M' in RG(N',M'B_{0B}) reachable via σ' such that M'(P\{P})=M(P\{p}) and σ' =σ. Thus no transition from T\$TB_{outB}$ is enabled in M'. Moreover, $|\sigma' P^{inP}|=|\sigma' P^{outP}|$. Thus no transition from TB_{outB} is enabled in M' either. Hence, M' is a deadlock marking in RG(N',M'B_{0B}). On the other hand, suppose M' is a deadlock marking in RG(N',M'B_{0B}). Let M be a marking of N such that M(P\{p})=M'(P\{p}) and M(p)=0. By similar argument, we can also show M∈RG(N, MB_{0B}).

Liveness: Suppose a transition t∈T is enabled in M∈RG(N, MB0B). Let M[t>MB1B in RG(N, MB_{0B}) and σ be a firing sequence for M. Then σt is a firing sequence for MB_{1B}, and there is a marking M'B_{1B}∈RG(N',M'B_{0B}) reachable via σ' such that σ'=σ. As a result, t is also enabled in some marking M' in RG(N',M'B_{0B}) in the path σ' from M'B_{0B} to M'B_{1B}. On the other hand, suppose t ∈T is enabled in M'∈RG(N', M'B_{0B}). By similar argument, we can also show that t is enabled in some M∈RG(N, MB_{0B}). As a result, a transition t∈T is enabled in RG(N,MB_{0B}) if and only if it is enabled in RG(N', M'B_{0B}).

We construct the Petri nets model for application composed of semantic Web services using following algorithm.

Algorithm 1 Map OWL-S process model to the Petri nets model for application composed of semantic Web services.

(1) For every Web service ontology, whose behavior is represented by OWL-S, the structure and operational semantics of composite WS process are captured; the structure and operational semantics of composite WS are described by OWL-S process models.

(2) Utilize resolution principle to resolve the existential quantifier and universal quantifier by using skolem function and kripke structure. Then resolve the inference rule of OWL-S process model.

(3) According to dependent relation of all Web services, OWL-S process are mapped to Petri nets, and Petri nets model for WS is constructed as a tree using definition 1.

(4) Then the Petri nets models for composition of all Web services are constructed by using definition 2-8. The nodes of the tree correspond to composite processes that represent different control constructs such as Choice for non-deterministic choices, Sequence for deterministic sequences of processes, and If conditionals. Atomic processes are represented as the leaves of the tree.

(5) For those services without preceding service, the transitions representing their beginning are combined to one transition, named as tb. Then, introduce a place s_0, such that $\bullet t_b=\{s_0\}$, $t_b\bullet=\{s_i$, where service i has no preceding service $\}$, $\bullet s_0=\varphi$, $s_0\bullet=\{t_b\}$, $W(s_0,t_b)=1$, $M_0(s_0)=1$.

(6) For those services without succeeding service, the transitions representing their accomplishment are combined to one transition, named as t_e. Then, introduce a place s_e, such that $\bullet t_e=\{s_i$, where service i has no succeeding service $\}$, $t_e\bullet=\{s_e\}$, $\bullet s_e=\{t_e\}$, $s_e\bullet=\varphi$, $W(t_e, s_e)=1$, $M_0(s_e)=0$.

In order to use Petri nets as a process model for control purposes, the analysis of their corresponding reachability graphs has turned out to be a suitable analysis technique. Algorithm 2 can generate the reachability and coverability graph.

Algorithm 2 Reachability and coverability graphs Generation.

Reachability-Graph(<P, T, F,W,M_0>)

```
1 <V, E, v0> := <{M0},Φ,M0>;
2 Work : set := {M0};
3 while Work ≠ Φ;
4 do select M from Work;
5    Work := Work \ {M};
6    for t ∈ enabled(M)
7    do M':= fire(M, t);
8       if M'∉V
9          then V:= V ∪ {M'}
10      Work := Work ∪ {M'};
11      E := E ∪{<M, t,M0>};
12 return <V, E, v0>;
```

The algorithm makes use of two functions: The set Work may be implemented as a stack, in which case the graph will be constructed in a depth-first manner, or as a queue for breadth-first. Breadth first search will find the shortest transition path from the initial marking to a given (erroneous) marking. Some applications require depth first search. (Huaikou Miao et al.,2008).

4. Colored Petri nets-based models for Web services composition

In this section, a colored Petri nets (CPN) (Jensen K.1994) based algebra for modeling Web services is proposed. The model is expressive enough to capture semantics of complex service combinations and their respective specificities.

The Web service is formally defined and the obtained framework enables declarative composition of Web services (Zhaoli Zhang, et al.2008). Within the model, availability, confidentiality and integrity of the composite service can be analyzed.

4.1 Web services as colored Petri nets

By a Web service we mean a software component that is described via WSDL and is accessible via standard network protocols such as but not limited to SOAP over HTTP. Web services should be based on open standards, platform independent, application independent, and enable to share data and resources. Web service composition is a task of combining and linking existing Web services to create new web processes in order to add value to the collection of services.

For the sake of fast computation, many researchers prefer Petri nets(Thomas J P,et al.2005),since they are well suited for capturing flows in Web services, modeling the distributed nature of Web services, representing methods in a Web service and reasoning about the correctness of the flows.

A Web service behavior is basically a partially ordered set of operations. Therefore, it is straightforward to map it into a Petri nets. Operations are modeled by transitions and the state of the service is modeled by places.

The arrows between places and transitions are used to specify causal relations. Web services can be categorized into material services (e.g., delivery of physical products), information services (create, process, manage, and provide information), and material information

services, the mixture of both. Therefore, information is modeled by tokens and the types of information are modeled by the colors of the tokens.

It is assumed that a Petri nets, which represents the behavior of a service, contains one input place (i. e., a place with no incoming arcs and one output place (i. e., a place with no outgoing arcs.A Petri nets with one input place for absorbing information, and one output place for emitting information, will facilitate the definition of the composition operators and the analysis as well as the verification of certain properties (e.g., reachability, availability, and security. At any given time, a Web service can be in one of the following states: NotInstantiated, Ready, Running, Suspended, or Completed(Schuster H,et al.2000). When a Web service is in the Ready state, it means that tokens in their corresponding input place enable postset (set of transitions) of input place to fire.

Whereas the Completed state means that preset (set of transitions) of output place has fired and has generated tokens in corresponding output place.

Definition 10 (Service net) SN=(P, T, F, C, I_-, I_+,M_0, i ,o, l) is called a service net if and only if: (i) Σ=(P, T, F, C, I_-, I_+, M_0) is a colored Petri nets: (ii) i∈P is the input place with $^\bullet$x=∅ that is $\neg\exists o\in T$: (t,i) ∈F; (iii) o∈P is the output place with $x^\bullet$=∅ that is $\neg\exists o\in T$: (t,i) ∈F; (iv) l:T→A∪τ is a labeling function where A is a set of operation names. It is assumed that τ ∉A denotes a silent operation.

Silent operations are transition firings that cannot be observed. They are used to distinguish between external and internal behavior of the services. The sample service net is shown in Fig.9. The incoming arcs of a transition are marked with the number and color of tokens to enable the transition, and the outgoing arcs are marked with number and color of tokens that the transition generates. The color of token represents the type of information but not the content of information. Thus, two tokens with same color may be different. Obviously, transition t1 is enabled, and this service net is in Ready state. Subsequently, when this service net reaches Completed state, the output place o will contain four tokens: two colored "a", one colored "e", and one colored "f".

Now we give a formal definition of a Web service.

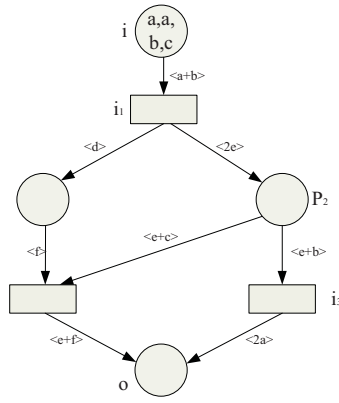


Fig. 9. Sample service net

Definition 11 (Web service) A Web service is a tuple S= (ID, Desc, Loc, URL, CS, SN), where ID is the unique identifier of the service which can be a name or a global unique number; Desc is the description of the service provided which summarizes what the service offer: Loc is the server in which the service is located:
URL is the invocation of the Web service: CS is a set of its component services. If CS=S.ID then S is a basic service. Otherwise S is a composite service; SN=(P, T, F, C, I_-, I_+, M_0, i, o, l) is the service net modeling the dynamic behavior of the service.

4.2 Composing Web services

Researchers have discussed various composite constructs (Narayanan S, Ren Z H, et al.,2003). In this chapter, we take sequence, concurrent, choice and loop constructs as basic constructs specified in the control flow, and take replace as advanes constrmct.We also give a formal semantics to the proposed algebra in terms of Petri nets.

4.2.1 Composite constructs

Below we describe syntax and informal semantics of the service algebra operators. The constructs are chosen to allow basic and advanced Web service composition.
The services can be defined as:
Definition 12 S::= ϵ | X | Seq(S,S) | Conc(S,S) | Choice(S,S) | Loop(S) | Rep(S,a,S), where ϵ represents an empty service, i.e., a service performs no operation.
X represents a service constant, used as an atomic or basic service in this context.
Seq(S_1,S_2) represents a composite service that performs the service S_1 followed by the service S_2. Seq(.) is an operator of sequence. If a composite service that performs either the service S_1 followed by the service S_2, or S_2 followed by S_1, it is called unordered sequence. In practice, we can decide the order by any condition since the order is not important, then unordered sequence could be treated as sequence.
Conc(S_1,S_2) represents a composite service that performs the services S_1 and S_2 independently. Both services are concurrently enabled and the overall composite service waits until both services are completed. Conc(.) is a concurrent operator.
Choice(S_1,S_2) represents a composite service that behaves as either service S_1 or service S_2. Once one of them is executed, the other is discarded. Choice(.) is a choice operator. The choice is not arbitrary (in fact, there is no absolute arbitrariness), and depends on conditions. Thus, the condition construct with Boolean variants could be treated as choice.
Loop(S) represents a composite service that performs a certain number of times of the service S. Loop(.) is a loop operator.
Rep(S_1, a, S_2) represents a composite service that behaves as S_1 except for operation in S_1 with label a that is replaced by the nonempty service S_2. Rep(·) is a replace operator.
The proposed algebra verifies the closure property. It guarantees that each result of an operation on services is a service to which one can again apply algebra operators. Software engineers thus are able to build more complex services by aggregating and reusing exfisting services through service algebra.

4.2.2 Formal semantics

Let S_i=(ID_i, $Desc_i$, Loc_i, URL_i, CS_i, SN_i) with SN_i=(P_i, T_i, F_i, C_i, I_i, I_{+I}, M_{0i}, i_i, o_i, l_i) for i=1,…,n, be n Web services such that $P_i \cap P_j = \varnothing$ and $T_i \cap T_j = \varnothing$ for $i \neq j$.

It is important to note that service composition, as will be described below, applies to syntactically different services. This is due to the fact that the places and transitions of the component services must be disjoint for proper composition (Zhaoli Zhang, et al.2008). However, a service may be composed with itself. In this case, the overlapping must be resolved prior to composition. This can be accomplished by renaming the sets P and T of one of the equal services. The two services remain equal up to isomorphism on the names of transitions and places. Note also that, in case of silent operations, we represent graphically the corresponding transitions as black rectangles.

Empty service The empty service ϵ is a service that performs no operation. It is used for technical and theoretical reasons.

Definition 13 The empty service is defined as ϵ =(ID, Desc, Loc, URL, CS, SN), where ID= Empty; Desc=Empty web service; Loc=Null, which means that there is no server for the service; URL=Null, which means that there is no URL for the service; CS={Empty} and SN=({p},∅,∅,C,0,0,{0}, p, p,∅).

Graphically, ϵ is represented by CPN of Fig.10(a) containing only one place.

Except for the empty service, in definitions below, ID is the ID of the new service; Desc is the description of the new service; Loc is the location of the new service; URL is the invocation of the new service.

Sequence The sequence operator allows execution of two services S_1 and S_2 in sequence. S_1 must be completed before S_2 can start. This is typically the case when a service depends on the output of the previous service.

Definition 14 The service Seq(S_1,S_2) is defined as Seq(S_1,S_2)=(ID, Desc, Loc, URL, CS, SN) where $CS=CS_1 \cup CS_2$, SN=(P, T, F, C, I-, I_+, M_0, i, o, l) where $P = P_1 \cup P_2$, $T = T_1 \cup T_2 \cup \{t\}$, $F = F_1 \cup F_2 \cup \{(o_1, t), (t, i_2)\}$, $i=i_1$, $o=o_2$, $I_- = I_- \cup I_-(o_1,t)$, $I_+=I_+ \cup I_+(i_2,t)$, $M_0=M_{01} \cup M_{02}$, and $l=l_1 \cup l_2 \cup \{(t,\tau)\}$.

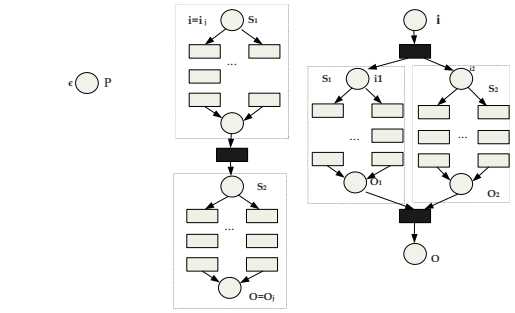
Given S_1 and S_2, Seq(S_1,S_2) is represented graphically by CPN shown in Fig.10(b). As mentioned above the second service depends on the output of the first service thus $C(i_2) \subseteq C(o_1)$ must be satisfied, or else, the second service cannot be enabled and the new composite service cannot work.

Concurrent The concurrent operator permits concurrent execution of two services S_1 and S_2. This is typically the case when some small (atomic) services without interfering with each other are merged into a bigger composite service.

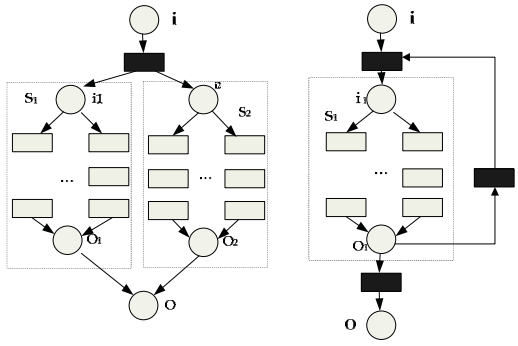
Definition 15 The service Conc(S_1,S_2) is defined as Conc(S_1,S_2)=(ID, Desc, Loc, URL, CS, SN) where $CS=CS_1 \cup CS_2$, SN={P, T, F, C,I_-,I_+,M_0,i, o, l) where $P = P_1 \cup P_2 \cup \{i, o\}$, $T = T_1 \cup T_2 \cup (t_i, t_o)$, $F= F_1 \cup F_2 \cup \{(i,t_i), (t_i,i_1), (t_i,i_2), (o_1,t_o), (o_2,t_o), (t_o,o)\}$, $I_-=I_- \cup I_-(i,t_i) \cup I_-(o_1,t_o) \cup I_-(o_2,t_o)$, $I_+=I_+ \cup I_+(i_1, t_i) \cup I_+(i_2,t_i) \cup I_+(o.t_o)$, $M_0=M_{01} \cup M_{02}$, and $l=l_1 \cup l_2 \cup \{(t_i, \tau), (t_o, \tau)\}$.

Given S_1 and S_2, Conc(S_1,S_2)is represented graphically by CPN as shown in Fig10(c).

Choice The choice operator performs either service S_1 or service S_2. Once one of them executes, another service is discarded.



(a) Empty (b) Sequence (c) Concurrent



(d) Choice (e) Loop

Fig. 10. Colored Petri nets of basic constructs

Definition 16 The service Choice(S_1,S_2) is defined as Choice(S_1,S_2)=(ID, Desc, Loc, URL, CS, SN) where CS =$CS_1 \cup CS_2$, SN=(P, T, F, C, I_, I_+,M_0,i, o, l) where P=$P_1 \cup P_2 \cup$ { i, o}, T = $T_1 \cup T_2 \cup$ {t_{i1}, t_{i2}, t_{O1},T_{O2}}, F=$F_1 \cup F_2 \cup$ { (i,t_{i1}), (i, t_{i2}), (t_{i1},i_1), (t_{i2},i_2), (o_1,t_{o1}),(o_2, t_{o2}), (t_{o1},o), (t_{o2},o)}, I_= I_ $\cup$ I_(i, t_{i1}) $\cup$ I_(I,t_{i2}) $\cup$ I_(o_1,t_{o1}) $\cup$ I_(o_2,t_{o2}), I_+=$I_+ \cup I_+$(i_1,t_{i1})$\cup I_+$(i_2,t_{i2})$\cup I_+$(o,t_{o1})$\cup$I+(o,t_{o2}), M_0=$M_{o1} \cup M_{02}$, and l=$l_1 \cup l_2 \cup$\{(t_{i1},τ), (t_{i2},τ), (t_{o1},τ), (t_{o2},τ)}.

Given S_1 and S_2, Choice(S_1,S_2) is represented graphically by CPN as shown in Fig.10(d).

Loop The loop operator allows that the service S performs a certain number of times. Typical examples where loop is required are communication and quality control where services are execrated more than once.

Definition 17 The service Loop (S_1) is defined as Loop(S_1)=(ID, Desc, Loc, URL, CS, SN), where CS=CS_1,SN=(P,T,F, C, I_, I_+, M_0, i, o, l) where P = $P_1 \cup$\{i, o}, T=$T_1 \cup$\{t_i,t_o,t}, F=$F_1 \cup$ {(i,t_i), (t_i,i_1), (o_1,t_o), (t_o,o), (o_1,t), (t,i_1)}, I_=I_$\cup$ I_(i, t_i) $\cup$I_(o_1,t_o) $\cup$I_(o_1,t), I_+=$I_+ \cup I_+$(i_1,t_i) $\cup I_+$(o,t_o) $\cup$ I_+(i_1,t), M_0=M_{01}, and l=$l_1 \cup$\{(l_i,τ),(t_o,τ),(l,τ)}.

Given S_1, Loop(S_1) is represented graphically by CPN as shown in Fig.10(e).

Replace The replace construct, in which operations are replaced by more detailed nonempty services, is used to introduce additional component services into a service.

Replace is the transformation of a design from a high level abstract form to a lower level more concrete form hence allowing hierarchical modeling.

Definition 18 Let $a \in A$. The service $Rep(S_1,a,S_2)$ is defined as $Rep(S_1,a,S_2)=(ID, Desc, Loc, URL, CS, SN)$.

If $a \in I_1(T_1)$ then $CS = CS_1 \cup CS_2$, otherwise $CS = CS_1$.

$SN=(P, T, F, C, I_-, I_+, M_0, i, o, l)$ where $P=P_1 \cup (P_2 - \{i_2,o_2\})$, $T=T_1 \cup T_2 - l_1^{-1}(a)$, $F=(F_1 - \{(x,y) \mid x \in l_1^{-1}(a) \text{ or } y \in l_1^{-1}(a)\}) \cup F_2$, $I_- =(I_{-1} - \{ I_{-1}(p,t) \mid t \cup I_1^{-1}(a)\}) \cup I_{-2}$, $I_+=(I_+ - \{ I_{+1}(p,t) \mid t \cup I_1^{-1}(a)\}) \cup I_{+2}$, $M_0=M_{01} \cup M_{02} - M(i_2) - M(o_2)$, if $t \cup (T_1 - I_1^{-1}(a))$ then $l(t)= l_1(t)$, otherwise $l(t)= l_2(t)$.

Given S_1, a and S_2, $Rep(S_1,a,S_2)$ is represented graphically by CPN shown in Fig.11. From the definition and Fig.11, we could find that the labeled transition, which to be replaced, should have only one incoming arc and one outgoing arc. If each transition in a Petri nets has only one incoming arc and one outgoing arc, the Petri nets is an ordinary Petri nets. We recommend high-level design is the ordinary Petri nets for allowing replace and hierarchical modeling.

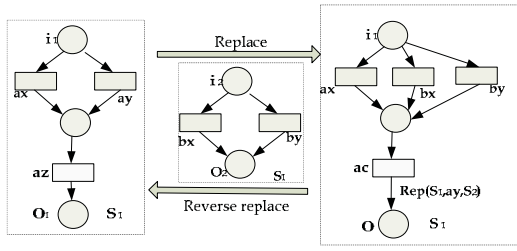


Fig. 11. Colored Petri nets of replace construct

4.3 Composing Web services

A composite Web service is a system that consists of several conceptually autonomous but cooperating units. It is difficult to specify how this system should behave and ensure that it behaves as required by the specification. However, the model based CPN could do something.

4.3.1 Closure property

As mentioned above, the proposed algebra verifies the closure property.

Theorem 4 The services compositions presented in Definition 12 are closed.

Proof The closure property of one-step composition is an immediate consequence of Definition 12, and the closure property of multiple-step composition can be proved by mathematical induction.

Not only it guarantees that each result of an operation on services is a service to which one can again apply algebra operators described in 4.2.1, but also it allows hierarchical modeling. For high-level abstract model, via replace, it can be transformed to lower-level model, which also can be transformed to higher-level model via reverse replace (see Fig.11). Behavioral equivalences are useful in verification as they lay the conceptual basis for

deciding that the behavior of two Web services can be considered to be "the same". They can also be used as a tool for reducing verification effort by replacing the CPN of a service by a smaller (in size) but "equivalent" one. Hence, analysts can analyze and verify the model in different levels.

4.3.2 Availability

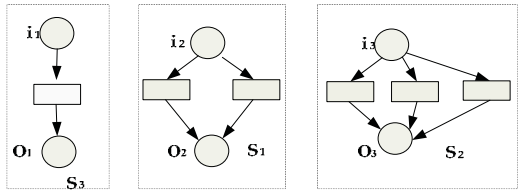
In a CPN model, the data (information) type is distinguishable, thus the designer can figure out which services can be composed together and which cannot. The data types are defined as classes, including ordinary types (such as integer, real, Boolean, etc.), same as in object-oriented programming. In a CPN model, colors of the tokens represent the classes (data types).

Since subclass inherits from superclass, color should be redefined to reflect inheritance. For instance, there are two superclasses: sc_1 and sc_2, and there are two subclasses: bc_1 inherits from sc_1, and bc_2 inherits from sc_2.

If sc_l is colored "a" and sc_2 is colored "b", then bc_l and bc_2 should be colored "a-c" and "b-d" to reflect the inheritance relation. Consequently, when a service input type requires color "a", both sc_l and bc_l are accept able. However, when a service input type requires color "a-c", only bc_l is acceptable and sc_l will be denied.

Consider three services as shown in Fig.12. We note that service S_1 cannot be composed with itself by loop construct because its output place cannot provide token colored "a", which is needed by its own input place I_1. For the same reason, service S_2 cannot be composed with itself by loop construct, but service S_3 can. We can also find that service S_1 can be composed with service S_2 by sequence construct only by order that S_2 is followed by S_1,and service S_1 can be composed with service S_3 by sequence construct only when S_1 is followed by S_3.

However, service S_2 can be composed with service S_3 by sequence construct by any order.



$C(i_1)=\{a\}, C(i_2)=\{b\text{-}f,c\}, C(i_3)=\{b,c\}$

$C(o_1)=\{b\text{-}d,c\}, C(o_2)=\{b\text{-}e,a,c\}, C(o_3)=\{b\text{-}f,c\}$

Fig. 12. Availability of composition

From the above analysis, we can reach the following conclusions:

If composite service $S = Seq(S_1,S_2)$ is available, $C(i_2) \subseteq C(o_1)$ must be satisfied.

If composite service $S = Conc(S_1,S_2)$ is available, the following must be satisfied: $C(i_1) \cup C(i_2) \subseteq C(i)$ and $C(o) = C(o_1) \cup C(o_2)$.

If composite service $S = Choice(S_1,S_2)$ is available, the following must be satisfied: $C(i_1) \cup C(i_2) \subseteq C(i)$ and $C(o) = C(o_1) \cup C(o_2)$.

If composite service $S = Loop(S_1)$ is available, $C(i_1) \subseteq C(o_1)$ must be satisfied.

In an elementary Petri nets, the data type is not distinguishable; hence the elementary Petri nets-based model for web composition cannot verify the availability feature of the composite service (Hamadi R,2003).

4.3.3 Security

Security is an important issue in information system. Confidentiality policies emphasize the protection of confidentiality (Zhaoli Zhang, et al.2008). Multilevel security (MILS) has a long tradition in military environments and is an important requirement in the trusted computer system evaluation criteria (TCSEC) for the security classes.

Subjects and objects of a system are assigned security classes (e.g. "high" and "low")with a specific order (high>low). A well known MILS model is the BellLaPadula model[ll]. The two most prominent rules are No-read-up and No-write-down which state that a low-level subject is not allowed to read high-level objects, and high-level objects can only be written by low-level subjects. These two rules result in an information flow from "low" to "high".

If the component services, transitions and places are assigned security level, the security feature of the Petri nets-based model for Web service composition can be verified by coverability graph(Knorr K.,2001).

The simplest type of confidentiality classification is a set of security clearances arranged in a linear (total) ordering. Let L(S) be the security clearance of service S. Expand the security clearances by adding a set of categories to each security classification. Each category describes a kind of information in all of those categories.

These sets of categories form a lattice under the operanon $\subseteq$ (subset of). Let G(S) be the category set of service S. Each security clearance and category forms a security level. Define the relation dom (dominates) as follows. (Bell D &Lapadula L.1996)

Definition 19 The security level (L,G) dom-inates the security level (L', G') if and only if $L' \leq L$ and $G' \subseteq G$.

Definition 20 (Simple security condition) S can read O if and only if S dom O, and S has discretionary read access to O.

To preserve confidentiality, following properties must meet while composing Web services。

Property 1. For composite service S, if S = Seq(S_1, S_2), S_2 dom S_1.

Proof If service $S_1$and S_2 can be composed with the sequence construct by the order that S1 is followed by S_2,then S_2 can read S_1. For satisfying Definition 20, this can happen if and only if S_2 dom S_1.

Property 2. For composite service S, if S = Conc(S_1,S_2), S_1 dom S, and S_2 dom S.

Proof If service S_1 and S_2 can be composed with the concurrent construct to form composite service S, then S_1 and S_2 can read S. For satisfying Definition 20, this can happen if and only if S_1 dom S, and S_2 dom S.

Property 3. For composite service S, if S=Choice(S_1,S_2),S_1 dom S, and S_2 dom S.

Proof If service S and S_2 can be composed with the choice construct to form composite service S, then S_1 and S_2 can read S. For satisfying Definition 20, this can happen if and only if S_1 dom S and S_2 dom S.

Integrity policies focus on integrity rather than confidentiality, because most commercial and industrial firms are more concerned with accuracy than disclosure. An integrity policy is a security policy dealing only with integrity. The strict integrity policy is most commonly called "Biba's Mode1", in which integrity labels are assigned to the objects and subjects in a

system. This model is the mathematical dual of the Bell-LaPadula model. Obviously, if the component services, transitions and places are assigned integrity level, the integrity feature of the Petri nets-based model for Web service composition can also be verified by coverability graph.

Let I(S) be the integrity level of service S. To preserve integrity, the following rules must be met while composing Web services:

For composite service S= Seq(S_1,S_2), $I(S_2) \le I(S_1)$.

For composite service S=Conc(S_1,S_2),$I(S_1) \le I(S)$, and $I(S_2) \le I(S)$.

For composite service S=Choice(S_1,S_2),$I(S_1) \le I(S)$, and $I(S_2) \le I(S)$.

Because the Biba's model is the mathematical dual of the Bell-LaPadula model, the above conclusions can be proved like their counterparts in Bell-Lapadula model

5. Related work

In the research related to Web services, several initiatives have been conducted with the intention to provide platforms and languages that will allow easy integration of heterogeneous systems. In particular, such languages as UDDI, WSDL, SOAP and part of DAML-S ontology (ServiceProfile and ServiceGrounding), define standard ways for service discovery, description and invocation (message passing). Some other initiatives such as BPEL4WS and DAML-S ServiceModel, are focused on representing service compositions where flow of a process and bindings between services are known a priori(Rao J & Dustdar S, 2005). Ontology-driven Web services composition is used to discover and assemble services into processes for easier and better quality workflow executions given increasing number and complexity of Web services(Budaka I,et al.2005).

Model	Petri type	Data type distinguishable	Auto composition	Additional message	Availability verification	Security analysis
CPWSC	Colored	Yes	Yes	No	Yes	Yes
Guo 2006	Colored	Yes	No (pre-defined rule and conditions needed)	No	Yes	Not mentioned
Qian 2006	Elementary	No	Yes	Yes	Yes	Not mentioned
Hamadi 2003	Elementary	No	Yes	No	No	Not mentioned

Table 1. Comparison of Petri nets-based models for Web service composition

Besides that, current solutions for Web service composition include web components, pi-calculus, Model check-ing/FSM and Petri nets(Milanovic N &Malek M.2004). In 2003, Hamadi proposed a Petri nets-based model for Web service composition (Hamadi R,2003), in which the data types cannot be distinguishable because an elementary Petri nets model is used. In a recent research, a CP-net model for Web service composition is proposed (Guo 2006) (Guo Yubin,et al.2006).However the rules and procedures of composition must be defined previously, and the services composition chain cannot be generated automatically without pre-defined conditions. In the message oriented activity based Petri nets model (Qian 2006) (Qian Zhuzhong, et al.2006),the Web service composition relies on messages,

which increase complexity of com-position. A comparison of Petri nets-based models for Web service composition is shown in Table 1, in which CPWSC represents the model proposed in this chapter.

6. Conclusion

In this chapter, we proposed Petri nets-based algebra for composing Web services. The formal semantics of the composition operations is expressed in terms of P/T Petri nets and CPNs by providing a direct mapping from each operator to Petri nets construction. In addition, the use of a formal model allows verification of closure, availability, and security properties and detection of inconsistencies both within and between services (Zhaoli Zhang, et al.2008) . There are other issues in B2B E-commerce which can be successfully addressed by extending the framework presented in this chapter. Further work will include a more thorough analysis of the field in addition to practical testing experiments with the methods.

7. References

Bell D, Lapadula L. (1996). The Bell-LaPadula model [J] *Journal of Computer Security,* 1996, 4(2-3): 239-263.

Budaka I, Aleman-Meza B, Zhang R, et al. (2005). Ontology-driven web services composition platform [C]*// Proceedings of IEEE International Conference on E-commerce Technology,* San Diego. Los Alamitos: IEEE Computer Society Press, 2005: 146-152.

Cordoso, J.; Sheth, A. (2002). Semantic e-Workflow Composition, *Technical Report,* LSDIS Lab, Computer Science, University of Georgia, July 2002

Dustdar S, Schreiner W. (2005). A survey on web services composition [J] *International Journal of Web and Grid Services.* 2005, 1(1): 1-30.

Guo Yubin, Du Yuyue, Xi Jianqing. (2006). A CP-net model and operation properties for web service composition [J]. *Chinese Journal of Computers,* 2006, 29(7):1067-1075 (in Chinese).

Hamadi R, Benatallah B.(2003). A Petri nets-based model for web service composition [C] *//Proceedings of the 14th Australasian Database Conference,* Adelaide. Darlinghurt: Australian Computer Society, 2003: 191-200.

Huaikou Miao, Tao He, Zhongsheng Qian,(2008). Modeling and Analyzing Composite Semantic Web Service Using Petri nets, *4th Workshop on Service-Oriented Applications, Integration and Collaboration (SOAIC'08),* Oct 24, 2008, Xi'an, China

J.P. Thomas, M. Thomas, and G. Ghinea,(2003). "Modeling of Web Services Flow", *Proceedings of the IEEE International Conference on ECommerce (CEC'03)*.

Jensen K. (1994). An introduction to the theoretical aspects of colored Petri nets [J]. *Lecture Notes in Computer Science.* 1994. 803: 230-272.

Knorr K. (2001). Multilevel security and information flow in Petri nets workflows [C]*//Proceedings of the 9th International Conference on Telecommunication Systems-Modeling and Analysis, Special Session on Security Aspects of Telecommunication Systems,* Dallas. Los Alamitos: IEEE Computer Society Press, 2001: 9-20.

Martin, D. et al. (2004). Bringing Semantics to Web Services: The OWL-S Approach, *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition,* San Diego, July 2004

Maurice ter Beek, Maurice; Bucchiarone, Antonio; Gnesi, Stefania; (2007). Web Service Composition Approaches: From Industrial Standards to Formal Methods Internet and Web Applications and Services, *2007. ICIW '07. Second International Conference* on 13-19 May 2007 Page(s):15 – 15 Digital Object Identifier 10.1109/ICIW. 2007.71

Milanovic N, Malek M. (2004). Current solutions for web service composition [J]. *IEEE Internet Computing,* 2004, 8(6): 51-59.

Narayanan S, Mcilraith S. (2003). Analysis and simulation of web services [J]. Computer Networks, 2003, 42(5): 675-693.

Qian Zhuzhong, Lu Sanglu, Xie Li. (2006). Automatic Composition of Petri nets based web services[J]. *Chinese Journal of Computers,* 2006, 29(7): 1057-1066 (in Chinese).

Rachid Hamadi, Boualem Benatallah, (2003). A Petri nets-based Model for Web Service Composition, *Proceedings of the Fourteenth Australasian database conference on Database technologies 2003,* ACM Press, Adelaide, Australia, February 1, 2003, pp.191-200.

Rao J, Su X. (2005). A survey of automated web service composition methods [J]. *Lecture Notes in Computer Science,* 2005, 3387: 43-54.

Ren Z H, Cao J N, Chan T S, (2003). Composition and automation of grid services [J]. *Lecture Notes in Computer Science,* 2003, 2834: 352-362.

S. Narayanan and S. Mcllraith. (2002). Simulation, Verification and Automated Composition of Web Services, *Proceedings of the 11th World Wide Web Conference,* Honolulu, HI, USA, July, 2002, pp.77-88.

Schuster H, Georgakopoulos D, Cichocki A, et al. (2000). Modeling and composing service-based and reference process-based multi-enterprise processes [J]. *Lecture Notes in Computer Science,* 2000, 1789: 247-263.

Thomas J P, Tomas M, Ghinea G. (2005). Modeling of web services flow [C]*//Proceedings of IEEE International Conference on E-commerce,* San Diego, California. Los Alamitos: IEEE Computer Society Press, 2005: 391-398

Zhang Jia, Chung Jenyao, Chang C.K., Kim S., (2004). "WS-Net: A Petri-net based specification model for web services", *Proceedings of the second IEEE International Conference on Web Servics,* IEEE Press, San Diego, California, USA, pp.420-427.

Zhaoli Zhang, Fan Hong, Haijun Xiao,(2008). A colored Petri nets-based model for web service composition,*Journal of Shanghai University (English Edition),* 2008,12(4):323–329

Zhijun Ding; Junli Wang; ChangJun Jiang; (2005). Semantic Web Service Composition Based on OWL-S. *Semantics, Knowledge and Grid, 2005. SKG '05.* First International Conference on Nov. 2005 Page(s):98 - 98

Petri Nets Applications

Edited by Pawel Pawlewski

ISBN 978-953-307-047-6
Hard cover, 752 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Petri Nets are graphical and mathematical tool used in many different science domains. Their characteristic features are the intuitive graphical modeling language and advanced formal analysis method. The concurrence of performed actions is the natural phenomenon due to which Petri Nets are perceived as mathematical tool for modeling concurrent systems. The nets whose model was extended with the time model can be applied in modeling real-time systems. Petri Nets were introduced in the doctoral dissertation by K.A. Petri, titled "„Kommunikation mit Automaten" and published in 1962 by University of Bonn. During more than 40 years of development of this theory, many different classes were formed and the scope of applications was extended. Depending on particular needs, the net definition was changed and adjusted to the considered problem. The unusual "flexibility" of this theory makes it possible to introduce all these modifications. Owing to varied currently known net classes, it is relatively easy to find a proper class for the specific application. The present monograph shows the whole spectrum of Petri Nets applications, from classic applications (to which the theory is specially dedicated) like computer science and control systems, through fault diagnosis, manufacturing, power systems, traffic systems, transport and down to Web applications. At the same time, the publication describes the diversity of investigations performed with use of Petri Nets in science centers all over the world.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Huaikou Miao and Tao He (2010). Petri Nets-based Models for Web Services Composition, Petri Nets Applications, Pawel Pawlewski (Ed.), ISBN: 978-953-307-047-6, InTech, Available from: http://www.intechopen.com/books/petri-nets-applications/petri-nets-based-models-for-web-services-composition

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821