Design of Service Net based Correctness Verification Approach for Multimedia Conferencing Service Orchestration

Cheng Bo*, Zhang Chengwen, Chen Junliang
1 State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications
2 Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing, 100876, China
E-mail: {chengbo, cwzhang, chjl}@bupt.edu.cn

Abstract

Multimedia conferencing is increasingly becoming a very important and popular application over Internet. Due to the complexity of asynchronous communications and handling large and dynamically concurrent processes for multimedia conferencing, which confront relevant challenge to achieve sufficient correctness guarantees, and supporting the effective verification methods for multimedia conferencing services orchestration is an extremely difficult and challenging problem. In this paper, we firstly present the Business Process Execution Language (BPEL) based conferencing service orchestration, and mainly focus on the service net based correction verification approach for multimedia conferencing services orchestration, which can automatically translated the BPEL based service orchestration into a corresponding Petri net model with the Petri Net Markup Language (PNML), and also present the BPEL service net reduction rules and multimedia conferencing service orchestration correction verification algorithms. We perform the correctness analysis and verification using the service net properties as safeness, reachability and deadlocks, and also provide an automated support tool for the formal analysis and soundness verification for the multimedia conferencing services orchestration scenarios. Finally, we give the comparison and evaluations.

Keywords: Service net, reduction rules, correctness verification, multimedia conferencing, services orchestration

1. Introduction

The ubiquity of Internet and IP network has made converged multimedia communication services over IP increasingly important and popular. Multimedia conferencing is increasingly becoming a very important and popular application. However, developing multimedia conferencing over those public networks is a challenging task, since a number of participants require scalable and flexible multimedia conferencing management that can deliver thousand or tens of thousands of simultaneous audio and video streams. The multimedia conference control and management are still in an early stage of development, and need to be extended in order to provide a fine grained management level that allows the deployment of multimedia conferencing solutions tailored to the user requirements and preferences, while maintaining high flexibility, scalability, and interoperability levels[1-2]. Current multimedia conferencing management systems can not fully address the problem of flexible and universal accessibility. Web service is an emerging trend in the industry today for providing distributed Internet services over IP networks and it relies on some structured XML based SOAP message and service WSDL to access, control and integrate various services remotely for complex transactions. Recent advances in Web services have made it practical to provide communication Web services [3-4], which is a new trend in the industry to enable communication through Service Oriented Architecture (SOA) and thereby package communications resources as services, and receiving
increasing attention, because it is well-suited for new, unified, converged communication initiatives and applications. The idea of using Web services to provide a standard interface for multimedia conferencing over Internet is very attractive, which will open a new paradigm of Web service based VoIP communication, and is extensible can be easily integrated in end to end SOA solutions, and also will have a significant impact on the evolution of open service marketplace. However, the design of Web services oriented multimedia conferencing process management, where a set of conferencing components services that interface existing communication capabilities must be organized and composed, which is commonly introduced to as Web services orchestration pattern. Multimedia conferencing services orchestration presents the different communication services which can be composed efficiently through a process flow in order to complete a conferencing process management. However, in particular, the composed multimedia conferencing services strongly rely on asynchronous communications to handle heterogeneous events generated by the network protocol resources, and they usually handle large and dynamically evolving sets of concurrent processes that realize the concrete multimedia conferencing process management [5-7]. The complexity of such mechanisms brings the great challenges and difficulties to achieve an effective correctness analysis and verification approach for the multimedia conferencing services orchestration.

Our work confronts both with the issue of patterns for the composed multimedia conferencing services orchestration, and effective approach for the correctness analysis and verification. Concerning the former, we present the multimedia conferencing service orchestration using standard BPEL language. Concerning the latter, we present the service net reduction based correctness analysis and verification approach for multimedia conferencing service orchestration, to address the problem of automated correctness verification, e.g. to verify the BPEL service net safeness issue, to detect the BPEL service net reachability, and to detect the BPEL service net deadlocks induced by critical runs. The remainder of the paper is organized as follows: Section 2, is the BPEL based multimedia conferencing services orchestration pattern. Section 3, is the Petri net and the BPEL transforming algorithms. Section 4, is the BPEL service net reduction rules. Section 5, is the correctness analysis and verification approach for multimedia conferencing services orchestration. Section 6, is the measurement and comparison. Section 7, conclusions and future work.

2. BPEL based Conference Service Orchestration

BPEL [8] is an orchestration language used to define a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners. The interaction with each partner occurs through Web Service interfaces, and the structure of the relationship at the interface level is encapsulated in what called a partner link. Using BPEL, those business processes which integrate a Web service collection in the same process flow can be constructed. As mentioned above, it is also possible to use BPEL to define an executable multimedia conference process. The logic and state of the process determine the nature and sequence of the Web Service interactions conducted at each conference partner, and thus the interaction protocols. In this sense, a BPEL based multimedia conference process definition provides and uses one or more WSDL services, and also provides the description of the behavior and interactions of a conference process instance relative to its conference partners and resources through Web Service interfaces. That is, BPEL defines the message exchange protocols followed by the multimedia conference process of a specific role in the interaction.

Multimedia conferencing allows the creation of a conference and the dynamic management of the participants and the media involved. When received the initial request from the conference owner, then invokes the create conference Web service to create a conference call and return a unique conference identifier. As soon as the first participant connects, the conference becomes active status. The duration of the conference is then measured starting from the moment the conference has become active. During the course of this process, the conference owner can also invoke the get conference information Web service to acquire the current conference status information. When finished these services invoking, the conference has been set up successfully. The other multimedia conferencing process is controlled by events handlers triggered by the external messages. During the conference session the application is able to add or remove a specific media stream to a single participant, e.g. adding a video
bidirectional stream to a participant that has an audio connection to the conference. This can be obtained by invoking the add media for participant and the delete media for participant Web service. And also, to retrieve information related to the conference and its status, by invoking get conference information and get participants information Web services.

For multimedia conferencing is a converged voice and data applications, which management involves both real-time communications and non-real-time information processing tasks. In our early works, we had developed multimedia conferencing management using Web services orchestration approach over public networks, and mainly focus on the communication Web services encapsulation and BPEL based multimedia conferencing process logics in a coarse grained Web service for the purposes of flexible multimedia conferencing process management [9-10]. Especially, a BPEL provides the capability to handle the conferencing asynchronous communications, suspend and resume execution upon events of services, and the message is routed to the existing conferencing process instances, and which support stateful collaboration between Web services in a standardized, implementation independent way. It is more suitable to orchestrate those communication services as conferencing process using BPEL, such as createConference, inviteChairman, setUserMedia, changeChairman, applySpeak, disconnectUser, endConference and conferenceNotification scenarios. Especially, we show a BPEL based multimedia conferencing notification service orchestration as an illustration, and the corresponding BPEL specification code is given in the appendix A.

3. Petri Net and BPEL Transforming Algorithm

Petri Nets [11] is a modeling formalism used for the analysis of a wide range of systems coming from different domains and characterized by situations of concurrency, synchronization, and conflict [12-13]. A Petri net is a particular kind of bipartite directed graphs populated by three types of objects. These objects are places, transitions, and directed arcs. Directed arcs connect places to transitions or transitions to places. In its simplest form, a Petri net can be represented by a transition together with an input place and an output place. This elementary net may be used to represent various aspects of the modeled systems. For example, a transition and its input place and output place can be used to represent a data processing event, its input data and output data, respectively, in a data processing system. In order to study the dynamic behavior of a Petri net modeled system in terms of its states and state changes, each place may potentially hold either none or a positive number of tokens. Tokens are a primitive concept for Petri nets in addition to places and transitions. The presence or absence of a token in a place can indicate whether a condition associated with this place is true or false. When there is at least one token in every place connected to a transition, we say that the transition is enabled. Any enabled transition may fire removing one token from every input place, and depositing one token in each output place.

**Definition 1. Petri Net**, a Petri net is a labeled Place/Transition net, i.e., a tuple Snet = (P, T, W, i, o, ℓ) where:

- P is a finite set of places,
- T is a finite set of transitions representing the operations of the service,
- F ⊆ (P × T) ∪ (T × P) is a set of directed arcs (flow relation),
- i is the input place with *i = {x ∈ P | T(x, i) ∈ W} = φ,
- o is the output place with *o = {x ∈ P | T(o, x) ∈ W} = φ,
- ℓ : T → Δ ∪ {τ} is a labeling function where Δ is a set of operation names.

Generally, a Web service behavior is considered as a partially ordered set of operations. Thus, it can be easily mapped into a Petri net. Especially, the transitions are used to model the operations and the places to model the state of the service. The arrows between places and transitions are used to define the process flow relations.

**Definition 2. Service Net**, a service net is a tuple:

S = (Sname, Sdes, Spro, Surl, Sorc, Snet), where,

- Sname is the name of the service and used as its unique identifier.
- Sdes is the description of the service and summarizes what the service offers.
- Spro is the server that the service is located in.
- Surl is the invocation of the Web service.
- Sorc is a set of its component services.

Snet = (P, T, F, i, o, ℓ), is the service net modeling the dynamic behavior of the service.

Here, a token in place i denotes a case (such as an instance of a Web service) that is ready to be started,
whereas a token in place $o$ denotes a case that has been completed.

Since the semantics of Petri nets is formally specified, and a BPEL service net can be obtained by mapping each BPEL process to a Petri net. We defined the BPEL’s main patterns based on previous reference works \[14\], in terms of BPEL service nets.

**Definition 3. BPEL Service Net**, the structure of a BPEL service net is defined as follows:

$$BPEL \models S | S_1 \neq S_2 | S_1 \otimes S_2 | S_1 \odot S_2 | m \rightarrow S_1, t \rightarrow S_2 | \lambda S$$

Especially, $\emptyset S$ represents a service constant, used as an atomic or basic service. Fig.1 shows how this can be mapped onto a service net.

\[ S_1 \neq S_2 \] represents BPEL's sequence pattern, which contains one or more activities that are performed sequentially, such as the Web services $S_1$ followed by $S_2$. Fig.2 shows how this can be mapped onto a service net.

\[ S_1 \otimes S_2 \] represents BPEL's flow pattern, which represents a flow that provides the capability for the parallel execution and synchronization for a more Web services, such as Web services $S_1$ parallel by $S_2$. Fig.3 shows how this can be mapped into a service net.

\[ S_1 \odot S_2 \] represents BPEL's pick pattern, which either waits on a message event or a timing event. Fig.5 shows how this can be mapped onto a service net.

\[ \lambda S \] represents BPEL's while pattern, which supports iterative performance of a specified iterative activity and is performed until the given boolean while condition no longer holds true. Fig.6 shows how this can be mapped onto a service net.

![Basic service net](image1)

**Fig.1. Basic service net**

![The service net for BPEL sequence pattern](image2)

**Fig.2. The service net for BPEL sequence pattern**

![The service net for BPEL flow pattern](image3)

**Fig.3. The service net for BPEL flow pattern**

![The service net for BPEL pick pattern](image4)

**Fig.4. The service net for BPEL switch pattern**

![The service net for BPEL pick pattern](image5)

**Fig.5. The service net for BPEL pick pattern**

![The service net for BPEL while pattern](image6)

**Fig.6. The service net for BPEL while pattern**
Here, a while activity contains one activity and executes this activity as long as an associated condition turns into “true”, and also a while activity supports structured loops. In Fig.6, activity X has a sub activity S that is performed multiple times as long as the while condition z turns and the loop will exit if the condition does not hold any more.

Beside those main structured activities, and BPEL includes other basic activities, such as invoke, receive, reply, assign, throw, compensate, and also includes the control flow constructs as event, fault and compensation handlers. We also present a comprehensive and rigorously defined mapping of BPEL constructs onto Petri net structures. Here, we give a corresponding transforming algorithm from BPEL to Petri nets with mentioned formal methods. The flow of the transforming algorithm is described as Fig.7.

1. Analysis of the multimedia conferencing service orchestration BPEL specification, and to identify the location of the <Process> element in the document, then use the Initiate and final to represent the <Process> and </Process> elements separately with the above-mentioned Petri model.
2. Invoke the transformActivity algorithm in step. 3 recursively map the sub-activities in the multimedia conferencing service orchestration BPEL specifications.
3. To map a BPEL activity with corresponding Petri pattern, if this activity contains other sub-activities (also including structured activities in the sub-activities), it is necessary to map the sub activity with recursively until to the basic activity firstly.
4. If there are unmapped sub-activities, return to Step 3, otherwise return to Step 5.
5. Add information for the constraint arc.

Here, the Petri Net Markup Language (PNML) [15] is an XML-based interchange format for Petri nets, which is used to describe the places, transitions and arcs for Petri net. It should take as input BPEL process definitions and translates them to Petri nets in PNML format. The transforming rules for the BPEL into PNML are shown as Fig.8.
pattern based language, and defines the mechanisms for specifying transformations on the XML data to convert it into other forms. Here, is rather proper for the transformation from BPEL to PNML.

4. BPEL Service Net Reduction Rules

The states graph method\cite{16} is a possible and effective method to analysis and correctness verification for a small sized service nets. However, the multimedia conferencing services orchestration can be a lengthy BPEL process for a Petri nets which have a large number of reachable markings or states. The use of state graph method also becomes difficult when the size of the BPEL based conferencing service net increases. Reduction rules can enable a service net to be transformed into another simpler Petri net, but continue to have certain properties of the initial net. The reduction rules are described as follows:

Rule.1: Sequence places reduction rule, there exist places $p_a$, $p_b \in P$, and transition $t \in \mathcal{T}$, $p_a$ and $p_b$ are the only predecessor and successor places for the transition $t$, and $k,m,n \geq 1$, and if transition $t$ meets the following conditions:
\begin{itemize}
  \item $p_a^* \{ t \}$ and $p_b^* \{ t \}$
  \item $W(p_a, t) = W(t, p_b) = k$
\end{itemize}

We can then reduce the transition $t$, then the places $p_a$ and $p_b$ can be merged into a new place $p_\prime$ and the corresponding weight of input and output is $m$ and $n$ for the place $p_\prime$. The sequence places reduction process is shown as Fig.9.

![Fig.9. Sequence places reduction rule](image)

The sequence places reduction process in Fig.9 also can be represented by the service net as follows:
\begin{itemize}
  \item $P_\prime = P \setminus \{ p \}$
  \item $T_\prime = T \setminus \{ t \}$
  \item $F_\prime = F \cup \{(s, t), (p, q) | s \in ^* p_a, q \in p_b^* \}$
  \item $\{(p_a, t), (t, p_b)\} \cup \{(s, p_a), (p_b, q) \}$
  \item $\{ s \in ^* p_a, q \in p_b^* \}$
\end{itemize}

Rule.2: Sequence transitions reduction rule, there exist transitions $t_a$ and $t_b \in \mathcal{T}$, and place $p \in P$, and $t_a$ and $t_b$ are the only predecessor and successor transitions for place $p$, and $k,m,n \geq 1$, and if the place $p$ meets the following conditions:
\begin{itemize}
  \item $t_a^* \{ p \} \land t_b^* \{ p \} \land p^* \{ t_a \} \land p^* \{ t_b \}$
  \item $W(t_a, p) = W(p, t_b) = k$
\end{itemize}

We can then reduce the place $p$, the transitions $t_a$ and $t_b$ can be merged into a new transition $t$, and the corresponding weight of input and output is $m$ and $n$ for the transition $t$. The process of sequence transitions reduction is shown as Fig.10.

![Fig.10. Sequence transitions reduction rule](image)

The sequence transitions reduction process in Fig.10 also can be represented by the service net as follows:
\begin{itemize}
  \item $P_\prime = P \setminus \{ p \}$
  \item $T_\prime = T \cup \{ t \}$
  \item $F_\prime = F \cup \{(s, t), (t, q) | s \in ^* t_a, q \in t_b^* \}$
  \item $\{(t_a, p), (t_b, p)\} \cup \{(s, t_a), (t_b, q) \}$
\end{itemize}

Rule.3: Parallel places reduction rule, there exist places $p_1, p_2, \ldots, p_k \in P$, $t_a$ and $t_b$ are the only predecessor and successor transitions for the places $p_1, p_2, \ldots, p_k$, and if the transition $t_a$ and $t_b$ meet the following conditions:
\begin{itemize}
  \item $p_1^* = p_2^* = \ldots = p_k^* \{ t_a \} \land p_1^* = p_2^* = \ldots = p_k^* \{ t_b \}$
  \item $\{ p_1, p_2, \ldots, p_k \} \subseteq t_a \land \{ p_1, p_2, \ldots, p_k \} \subseteq t_b$
  \item $W(t_a, p_1) = i_1, \ldots, W(t_a, p_k) = i_k$
  \item $W(t_b, p_1) = o_1, \ldots, W(t_b, p_k) = o_k$
\end{itemize}

We can then merge the places $p_1, p_2, \ldots, p_k$ into a new place $p$, and the corresponding weight of input and output is $i$ and $o$ for place $p$. The parallel places reduction process is shown as Fig.11.

![Fig.11. Parallel places reduction rule](image)

The parallel places reduction process in Fig.11 also can be represented by the service net as follows:
\begin{itemize}
  \item $P_\prime = P \setminus \{ p_1, p_2, \ldots, p_k \}$
  \item $T_\prime = T$
\end{itemize}
\[ F' = F \cup \{ (t_a, p) \mid (p, t_b) \} \rangle \]
\[ \{ (t_a, p_i) \mid (p_i, t_b) \} \} = 1, 2, \ldots, k \}
\[ W(t_a, p) = i = i_1 + i_2 + \ldots + i_k, \]
\[ W(p, t_b) = o = o_1 + o_2 + \ldots + o_k \]

**Rule 4: Parallel transitions reduction rule**, there exist transitions \( t_1, t_2, \ldots, t_k \in T \), and the places \( p_a \) and \( p_b \) are the only predecessor and successor places for the transitions \( t_1, t_2, \ldots, t_k \), and if the places \( p_a \) and \( p_b \) meet the following conditions:

- \( t_1 = \ldots = t_k = \{ p_a \} \land \ t_1 = \ldots = t_k = \{ p_b \} \)
- \( \{ t_1, t_2, \ldots, t_k \} \subseteq p_a \land \{ t_1, t_2, \ldots, t_k \} \subseteq *p_b \)
- \( W(p_a, t_1) = i_1, \ldots, W(p_a, t_k) = i_k \)
- \( W(t_1, p_b) = o_1, \ldots, W(t_k, p_b) = o_k \)

We can then merge the transitions \( t_1, t_2, \ldots, t_k \) into a new transition \( t \), and the corresponding weight of input and output is \( i \) and \( o \). The parallel transitions reduction process is shown as Fig.12.

![Fig.12. Parallel transitions reduction rule](image)

The parallel transitions reduction process in Fig.12 also can be represented by the service net as follows:

- \( P' = P \)
- \( T' = T \cup \{ t \} \}
- \( F' = F \cup \{ (p_a, t) \mid (t, p_b) \} \}
- \( \{ (p_a, t_i) \mid (t_i, p_b) \} \} = 1, 2, \ldots, k \)
- \( W(p_a, t) = i = i_1 + i_2 + \ldots + i_k \)
- \( W(t, p_b) = o = o_1 + o_2 + \ldots + o_k \)

**Rule 5: Overlapping places reduction rule**, there exist

- \( P_1 = \{ p_1, p_2, \ldots, p_k \} \subseteq P \), and
- \( T_a = \{ t_1, t_2, \ldots, t_{m_a} \} \subseteq T \), \( T_b = \{ t_1, t_2, \ldots, t_{m_b} \} \subseteq T \), and places \( p_1, p_2, \ldots, p_k \) are the overlapping places, and if the transitions \( T_a \) and \( T_b \) meet the following conditions:

- \( \forall t \in T_a: t = p \cdot T_a = \{ t_1, t_2, \ldots, t_{m_a} \} \land \)
- \( p = \{ t_1, t_2, \ldots, t_{m_b} \} \)
- \( \forall t \in T_a: t = p \cdot T_b = \{ t_1, t_2, \ldots, t_{m_b} \} \)
- \( \forall t \in T_b: t = p \cdot T_a = \{ t_1, t_2, \ldots, t_{m_b} \} \)
- \( \forall t \in T_b: t = p \cdot T_a = \{ t_1, t_2, \ldots, t_{m_b} \} \)

We can then merge the places \( p_1, p_2, \ldots, p_k \) into a new place \( p \). The overlapping places reduction process is shown as Fig.13.

![Fig.13. Overlapping places reduction rule](image)

The overlapping places reduction process in Fig.13 also can be represented by the service net as follows:

- \( P' = P \)
- \( T' = T \cup \{ t \} \}
- \( F' = F \cup \{ (t, p), (p, t') \} \}
- \( \{ (t, p), (p, t') \} \} \subseteq T_a, t' \subseteq T_b \)

**Rule 6: Overlapping transitions reduction rule**, there exist

- \( \exists T_1 = \{ t_1, t_2, \ldots, t_k \} \subseteq T \), and
- \( \exists P_a = \{ p_1, p_2, \ldots, p_m \} \subseteq P \), and
- \( P_b = \{ p_1, p_2, \ldots, p_m \} \subseteq P \), and if the transitions \( P_a \) and \( P_b \) meet the following conditions:

- \( \forall t \in T_1: t = p_a \cdot T_1 = \{ p_1, p_2, \ldots, p_m \} \land \)
- \( t_p = p_b \cdot T_b = \{ p_1, p_2, \ldots, p_m \} \)
- \( \forall p \in P_a: p = p_1 \cdot T_1 \)
- \( \forall p \in P_b: p = p_1 \cdot T_1 \)

We can then merge the transitions \( t_1, t_2, \ldots, t_k \) into a new transition \( t \). The overlapping transitions reduction process is shown as Fig.14.

![Fig.14. Overlapping transitions reduction rule](image)

The overlapping transitions reduction process in Fig.14 also can be represented by the service net as follows:

- \( P' = P \)
- \( T' = T \cup \{ t \} \}
- \( F' = F \cup \{ (t, p), (p, t') \} \}
- \( \{ (p, t), (t, p') \} \} \subseteq T_a, t' \subseteq T_b \)
Rule 7: “OR” branch transition reduction rule, there exist \( p_1, p_2 \in P \), and \( t_1, t_2, t_3, t_4 \in T \), and if the places as \( P_1 \), \( P_2 \) and transitions as \( t_1, t_2, t_3, t_4 \) meet the following conditions:

- \( \forall p \in P, \exists t \in T \) such that \( p \rightarrow t \)
- \( p_1 \rightarrow t_1 \) and \( p_2 \rightarrow t_2 \)
- \( p_1 \rightarrow t_3 \) and \( p_2 \rightarrow t_4 \)

We can then reduce the place \( p_2 \), and merge the transitions \( t_1, t_2, t_3, t_4 \) into new transitions \( T_{23}, T_{24} \). The “OR” branch transition reduction process is shown as Fig. 15.

![Fig. 15. “OR” branch transition reduction rule](image1)

The “OR” branch transitions reduction process in Fig. 15 also can be represented by the service net as follows:

- \( P' = P \setminus \{ p_2 \} \)
- \( T' = T \cup \{ t_{23}, t_{24} \} \)
- \( F' = F \cup \{(t_{23}, p_1), (t_{24}, p_2)\} \setminus \{(t_1, p_1), (t_2, p_2), (t_3, p_1), (t_4, p_2)\} \)

Rule 8: “OR” merge place reduction rule, there exist \( p_1, p_2 \in P \), and \( t_1, t_2, t_3, t_4 \in T \), and if the places as \( P_1 \), \( P_2 \) and transitions as \( t_1, t_2, t_3, t_4 \) meet the following conditions:

- \( p_1 \rightarrow t_1 \) and \( p_1 \rightarrow t_4 \)
- \( p_2 \rightarrow t_2 \) and \( p_2 \rightarrow t_3 \)
- \( t_1, t_2, t_3, t_4 \) meet the following conditions:

We can then reduce the place \( p_1 \), and merge the transition \( t_1, t_2 \) and \( t_3, t_4 \) into new transition \( t_{23} \) and \( t_{24} \). The “OR” merge place reduction process is shown as Fig. 16.

![Fig. 16. “OR” merge place reduction rule](image2)

The “OR” merge place reduction process in Fig. 16 also can be represented by the service net as follows:

- \( P' = P \setminus \{ p_1 \} \)
- \( T' = T \cup \{ t_{23}, t_{24} \} \)
- \( F' = F \cup \{(t_{23}, p_1), (t_{24}, p_2)\} \setminus \{(t_1, p_1), (t_2, p_2), (t_3, p_1), (t_4, p_2)\} \)

Rule 9: “AND” branch transition reduction rule, there exist transitions \( t_1, t_2 \in T \), and \( p_1, p_2, p_3, p_4 \in P \), and if the places as \( P_1 \), \( P_2 \), \( P_3 \), \( P_4 \) and transitions as \( t_1, t_2 \) meet the following conditions:

- \( t_1 \rightarrow p_1 \) and \( t_2 \rightarrow p_4 \)
- \( t_1 \rightarrow p_3 \) and \( t_2 \rightarrow p_4 \)

We can then reduce the place \( p_2 \), and merge the places \( p_1, p_2, p_3, p_4 \) into new places \( p_{23}, p_{24} \). The “AND” branch transition process is shown as Fig. 17.

![Fig. 17. “AND” branch transition reduction](image3)

The “AND” branch transition process in Fig. 17 also can be represented by the service net as follows:

- \( P' = P \setminus \{ p_{23}, p_{24} \} \)
- \( T' = T \setminus \{ t_2 \} \)
- \( F' = F \cup \{(t_{23}, p_1), (t_{24}, p_2)\} \setminus \{(t_1, p_1), (t_2, p_2), (t_3, p_3), (t_2, p_4)\} \)

Rule 10: “AND” merge place reduction rule, there exist transitions as \( t_1, t_2 \in T \), and \( p_1, p_2, p_3, p_4 \in P \), and if the places as \( P_1 \), \( P_2 \), \( P_3 \), \( P_4 \) and transitions as \( t_1, t_2 \) meet the following conditions:

- \( p_1 \rightarrow t_1 \) and \( p_2 \rightarrow t_2 \)
- \( t_1 \rightarrow p_3 \) and \( t_2 \rightarrow p_4 \)

We can then reduce the place \( p_1 \), and merge the places \( p_1, p_2, p_3, p_4 \) into new places \( p_{14}, p_{24} \). The “AND” merge place reduction process is shown as Fig. 18.
Fig.18. “AND” merge place reduction rule

The “AND” merge place reduction process in Fig.18 also can be represented by the service net as follows:

\[
P' = P \setminus \{p_{14}, p_{24}\} \\
T' = T \setminus \{t_2\} \\
F' = F \cup \{(p_{14}, t_2)(p_{24}, t_2)\} \\
\{(p_1, t_1)(p_2, t_1)(t_1, p_4)(p_4, t_2)\}
\]

Rule.11: Cycle place reduction rule, there exist transition \( t \in T \), if the transition \( t \) meets the following condition:

1. \( \text{t} = \text{t}' \)
2. \( |\text{t}| = |\text{t}'| = 1 \)

Fig.19. Cycle place reduction rule

We can then reduce the place \( p \). The cycle place reduction process in Fig.19 also can be represented by the service net as follows:

\[
P' = P \setminus \{p\} \\
T' = T \setminus \{t\} \\
F' = F \setminus \{(p, t)(t, p) | p \in t\}
\]

Rule.12: Cycle transition reduction rule, there exist place \( p \in P \), and if the place \( p \) meets the following conditions:

1. \( \text{p} = \text{p}' \)
2. \( |\text{p}| = |\text{p}'| = 1 \)
3. \( M_\phi(p) \neq \emptyset \)

Fig.20. Cycle transition reduction rule

We can then reduce the transition \( t \). The cycle transition reduction process in Fig.20 also can be represented by the service net as follows:

\[
P' = P \setminus \{p\} \\
T' = T \\
F' = F \setminus \{(p, t)(t, p) | t \in p'\}
\]

5. Conference Services Orchestration Verification

Multimedia conferencing has high performance and availability requirements on Web services when they are deployed in the real network environment. These requirements are grounded on the correctness of composed conferencing service orchestration processes. We perform the correctness analysis and verification based on the service net properties for the BPEL based conferencing service orchestration processes. In this section, we focus on these properties: safeness, reachability and deadlocks.

The service net is safe only if each place is safe, and a place is safe if the token count does not exceed 1 in any marking. The coverability tree algorithm 1 is applied to verify the safeness for the above multimedia conferencing services orchestration process.

Algorithm 1 Coverability Tree Algorithm

Step 1) Label the initial marking \( M_0 \) as the root and tag it “new”.
Step 2) While “new” markings exists, do the following:
Step 2.1) Select a new marking \( M \).
Step 2.2) If “new” markings exists, do the following:
Step 2.2.1) Select a new marking \( M \).
Step 2.2.2) On the path from the root to \( M \), mark \( M \) as “dead-end”.
Step 2.2.3) If no transitions are enabled at \( M \), mark \( M \) as “old”.
Step 2.3) If there exists enabled transitions at \( M \), do the following for each enabled transition \( t \) at \( M \):
Step 2.3.1) Obtain the marking \( M' \) that results from firing \( t \) at \( M \).
Step 2.3.2) While there exists enabled transitions at \( M' \), do the following for each enabled transition \( \omega \) at \( M' \):
Step 2.3.2.1) Obtain the marking \( M" \) that results from firing \( \omega \) at \( M' \).
Step 2.3.2.2) On the path from the root to \( M" \) if there exists a marking \( M" \) such that \( M(p) = M"(p) \) for each place \( p \) and \( M"(p) \neq M" \), i.e., is coverable, then replace \( M"(p) \) by \( \omega \) for each \( p \), such that \( M"(p) \neq M"(p) \).
Step 2.3.3) Introduce \( M" \) as a node, draw an arc with label \( \omega \) from \( M \) to \( M" \), and mark \( M" \) as “new”.

Here, safeness is indicated by the fact that in the coverability tree there are no markings with elements exceeding one. Given a service net \( (N, M_0) \), from the initial marking \( M \), we can obtain as many “new” markings as the number of the enabled transitions. From each new marking, we can again reach more markings. This process results in a tree representation of the
markings. Nodes represent markings generated from $M_0$ and its successors, and each arc represents a transition firing, which transforms one marking to another. To keep the tree finite, we introduce a special symbol $\omega$, which can be thought of as “infinity”. Especially, if the coverability tree contained also leaves with markings different from the final, it would mean that there is a deadlock within the service net and the service net is live and level 1. If the coverability tree contained only leaves with markings different from final one, it would mean that the service net is live at level 0, i.e. is dead.

The reachability is a fundamental basis for studying in the dynamic properties of any service net. A marking $M$ is reachable if it is the marking reached by some occurrence sequence. The incidence matrix & state equation algorithm [18-19] is used to analysis the reachable issue. The following algorithm 2 gives the corresponding main steps.

Here, the firing of an enabled transition will change the token marking in a net according to the firing rule. A sequence of firings will result in a sequence of markings. A marking $M_n = (0, 0, 0, \ldots, 1)$ is said to be reachable from a marking $M_0 = (1, 0, 0, \ldots, 0)$ if there exists a sequence of firings that transforms $M_0$ to $M_n$. The reachability decision expression is as $M_n = M_0 + x \cdot A$, and where $A$ is the incidence matrix, $M_0 = [1, 0, 0, \ldots, 0]_{1 \times n}$, and $M_n = [0, 0, 0, \ldots, 1]_{1 \times n}$. If the $x$ can be solved, then the model is reachable.

It is important to analyze and detect the deadlock before the BPEL service net is put into operation [20]. The deadlock is an important correctness property of BPEL service net, which denotes the correctness of a BPEL based Web services orchestration logic. From the definition, it is obvious that if a BPEL based multimedia conferencing Web service orchestration logic is correct, it should not contain a deadlock. Especially, the deadlock for a BPEL service net is the problem of deciding if any of its reachable markings is a deadlock. If any transition in a BPEL service net can not be invoked, and then it belongs to a deadlock status, which means the conflict place in the BPEL service net, which describe the status of a token which is invoked from a place, through the transition, and then to another place, if marking of a net is a deadlock if it enables no transitions. Therefore, the deadlock problem of a BPEL based multimedia conferencing service orchestration can be verified by reducing its BPEL service net representation based on the above mentioned BPEL service net reduction rules.

The multimedia conferencing service orchestration process contains some real-time communication services, such as multiple party call control service and short message service, and also contains some non real-time Web services, such as user authentication and charging service. As a result, the multimedia conferencing server must provide the capability to execute those hybrid services together. We have developed SOA based multimedia conferencing system. The framework is shown as Fig.21.

![Fig.21. Conference notification services orchestration](image)

When the chairman sends a request to create conference to the multimedia conference server, which deployed the verified BPEL based multimedia conferencing process, and then begins to receive the request messages, and transfers the SOAP messages into normalized messages, and then routed the normalized message to
the BPEL engine and invokes the BPEL flow to create conference which has been compiled. After some necessary process, BPEL engine invokes Web service provided by external communication server, which interacts with media server by SIP signaling. Finally, RTP or RTCP media channel is established between the participants and the media server.

Multimedia conferencing services orchestration process specification files, including BPEL files and WSDL files, are input into transformation engine, which is in charge of execution of meta-model transformation rules, and are transformed to Petri net model described in output PNML files. Transformation engine is actually an XSLT processor, its function comprises not only the basic transformation but also later refinement operations, for instance, communication places combining. Then, PNML files are loaded into Petri net engine for simulation and verification. It provides an infrastructure for bringing ideas for analyzing, for simulating, or for verifying Petri Nets into action. The flow for the analysis and correctness verification approach for BPEL based multimedia conferencing services orchestration is shown as Fig.22.

It is observed from Fig.22, the service net based correctness verification algorithms are used in the multimedia conferencing service orchestration process, such as the coverability tree algorithm is applied to detect the safeness for the multimedia conferencing service orchestration flow, the incidence matrix & state equation algorithm is applied to find the reachability problem for the multimedia conferencing services orchestration, and a BPEL service net reduction rules approach is applied to detect the deadlock problems for the multimedia conferencing services orchestration process.

Especially, we choose to describe the formal Web services orchestration of the multimedia conferencing informing process for the participants using the Petri nets. First, it is important to decide which way should be adopted to inform the participants, and the multimedia conferencing informing process can be done by invoking an Email service, or a short message service, even a telephone service. The informing way can be achieved by invoking the informing way service. Especially, a participant would be informed by using a short message service or an Email service. If both the Email and short message informing services fail, and finally, there is a need to inform the participants by the telephone service. The Petri net based multimedia conferencing informing service orchestration is shown Fig.23.

Based on the transformation and reduction rules presented in section 3 and section 4, we can also get the corresponding service net for the multimedia conferencing informing service orchestration scenario. Here, we demonstrate the service net reduction based correction verification approach for the multimedia conferencing informing service orchestration scenario. Based on the transformation and reduction rules presented in section 3 and section 4, we can also get the corresponding service net for the multimedia conferencing informing service orchestration is illustrated as Fig.24.
Once we get the logic representation of multimedia conferencing informing services orchestration process, its correctness can be verified by applying the BPEL service net reduction rules. Fig. 24 shows the main steps to reduce the conferencing informing service orchestration logic to a single place, which indicates that the logic of conferencing informing service orchestration is correct.

### 6. Measurement and Comparison

This section presents the experimental results aimed at validating the feasibility of applying the presented correctness analysis and verification approach and traditional state graph reduction method in terms of execution time. To compare the efficiency of two different methods, we also implemented a graph reduction algorithm. The measurements have been run on a laptop with a Intel(R) Core(TM)2 Duo CPU, 1.58 GHz, 2 GB RAM, running Windows XP SP2. Here, the execution time for the correctness verification is measured for the complete BPEL processes for the multimedia conferencing services orchestration scenarios, namely `createConference`, `conferenceNotification`, `inviteChairman`, `setUserMedia`, `changeChairman`, `applySpeak`, `disconnectUser` and `endConference`, and so on. Here, the size of a BPEL process is measured in terms of the number of activities defined in the process. The size of a Petri net or a service net is measured in the number of places and transitions. We consider the scale of multimedia conferencing services orchestration, i.e. the number of places and transitions, is 0, 50, 150, 200, 250, 300, 350, 400, 450 and 500 respectively. The corresponding execution time for the verification is shown in Fig. 25.

![Fig. 24. Verification for conferencing informing service](image)

It is observed from Fig. 25, when the number of places and transitions is less than 100, the execution time of both verification approaches is almost similar. When the number of places and transitions is not more than 200, both the execution time for the verification is within

![Fig. 25. Verification for conferencing notification service](image)
25ms. When the number of places and transitions is equal to 500, the verification time of our proposed method is 250ms, which is only half of the verification time of the graph reduction method. Here, the graph reduction algorithm scans the BPEL service net and computes the relationship among transitions and places every time, while our proposed correctness verification algorithm operates on directly. The results have shown that the execution time for our proposed correctness verification approach decreases significantly than the execution time consumed by the graph reduction method.

7. Conclusions

Multimedia conferencing is increasingly becoming a very important and popular application over Internet. we firstly present the BPEL based conferencing service orchestration, and mainly focus on the service net based correction verification approach for multimedia conferencing services orchestration, which can automatically translated the BPEL based service orchestration into a corresponding Petri net model with the PNML, and also present the BPEL service net reduction rules and multimedia conferencing service orchestration correction verification algorithms. We perform the correctness analysis and verification using the service net properties as safeness, reachability and deadlocks, and also provide an automated support tool for the formal analysis and soundness verification for the multimedia conferencing services orchestration scenarios. In the future work, we will analyze the possible data dependency conflicts in multimedia conferencing services orchestration processes. Furthermore, the proposed BPEL service net reduction rules still need to be further tested on more complicated multimedia conferencing services orchestration cases.

Acknowledgements

This study is supported by National Natural Science Foundation of China (Grant No. 61001118, 60872042); “973” program of National Basic Research Program of China (Grant No. 2011CB302704). Fundamental Research Funds for the Central Universities (2011RC0203)

References

14. Chun Ouyanga, Eric Verbeekb, Wil M.P. van der Aalst,a,b, Stephan Breutela, Marlon Dumasa, Arthur H.M. ter Hofstede, Formal semantics and analysis of control


Appendix A. BPEL specification code for multimedia conferencing notification service orchestration

```xml
<bpel:process>
  <bpel:sequence>
    <bpel:receive createInstance="yes" operation="createConference" partnerLink="createConference"
      portType="ns1:CreateConference" variable="createConferenceRequest"/>
    <bpel:invoke inputVariable="createConferenceRequest" operation="createConference"
      outputVariable="createConferenceResponse" partnerLink="createConference" portType="ns1:CreateConference"/>
    <bpel:invoke inputVariable="getConfStatusRequest" operation="getConfStatus" outputVariable="getConfStatusResponse"
      partnerLink="getConfStatus" portType="ns1:GetConfStatus"/>
    <bpel:if>
      <bpel:condition>$getConfStatusResponse.parameters/ns1:getConfStatusReturn="initial"</bpel:condition>
      <bpel:sequence>
        <bpel:repeatUntil>
          <bpel:condition/>
          <bpel:flow>
            <bpel:repeatUntil>
              <bpel:condition/>
              <bpel:sequence>
                <bpel:invoke inputVariable="getUserStatusRequest" operation="getUserStatus"
                  outputVariable="getUserStatusResponse" partnerLink="getUserStatus" portType="ns1:GetUserStatus"/>
                <bpel:assign/>
                <bpel:invoke inputVariable="inviteUserRequest" operation="inviteUser" outputVariable="inviteUserResponse"
                  partnerLink="inviteUser" portType="ns1:InviteUser"/>
              </bpel:sequence>
            </bpel:flow>
          </bpel:repeatUntil>
        </bpel:sequence>
      </bpel:repeatUntil>
    </bpel:if>
    <bpel:if>
      <bpel:condition>$getConfStatusResponse.parameters/ns1:getConfStatusReturn="active"</bpel:condition>
      <bpel:sequence>
        <bpel:invoke inputVariable="getUserStatusRequest" operation="getUserStatus"
          outputVariable="getUserStatusResponse" partnerLink="getUserStatus" portType="ns1:GetUserStatus"/>
        <bpel:if>
          <bpel:condition>$getUserStatusResponse.parameters/ns1:getUserStatusReturn = "haschairmen"</bpel:condition>
          <bpel:if>
            <bpel:condition>"chairman is in conf"</bpel:condition> <!--assign success to createConferenceReturn-->
          </bpel:if>
        </bpel:if>
      </bpel:sequence>
    </bpel:if>
    <bpel:else> <!--assign failure to createConferenceReturn-->
      <bpel:if>
        <bpel:else> <!--assign failure to createConferenceReturn-->
          <bpel:else>
            <bpel:sequence>
              <bpel:reply operation="createConference" partnerLink="createConference" portType="ns1:CreateConference"
                variable="createConferenceResponse"/>
            </bpel:sequence>
          </bpel:else>
        </bpel:else>
      </bpel:else>
    </bpel:else>
  </bpel:sequence>
</bpel:process>
```