Verifying Web Services Choreography using Dead Marking (Deceased Smearing) Algorithm

N. Danapaquiame¹ Dr. E. Ilavarasan²

¹Research Scholar, Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry-605014, India, n.danapaquiame@gmail.com
²Associate Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry-605014, India.eilavarasan@pec.edu

Abstract. Internet is a network of interlinked computer networking worldwide, which is accessible to the general public. Web services are a method of communication between two electronic devices over the web (internet). The W3C defines a “web service” as a software system designed to support interoperable machine to machine interaction over the “network”. Web service composition provides an open, standard – based approach for connecting web services together to create higher level business processes. This web service composition turns out to be a major research issues. I.e. Only limited number of verification approaches is available. The existing approaches concentrate mainly on composition and not on verification. Thus, there is a need for verification of web service composition. This paper proposes a new model called Buchi automata and introducing a new algorithm called dead marking algorithm (Deceased smearing ) for verifying the composed web services. It identifies whether the state is in dead or not.

Keywords: Buchi Automata, Dead Marking. W3C.

1 Introduction

The term Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. Web Services are therefore applications that provide services that can be obtained through the Internet. Service oriented architecture (SOA) is an architectural strategy that enables Web applications to be built using services and one way of implementing this architecture is by using Web services. Basically, a service provider develops a service and publishes the service description in a service registry. In general, a service client to find the required service from the service registry sends a request to the service provider via SOAP and receives a response in return. The type of service could be such as retrieving available seat information in a transport service or buying stationary items from a Web store. However, not all service is able to satisfy the needs of a user. In this situation, it is possible to combine existing services together in order to fulfill this need. The act of combining these services is called Web service composition (WSC). Although many researchers have been focused on
the discovery, selection and composition of Web services, research areas such as verification of Web services especially in Web service composition is still new and immature.

In recent years, many Web service composition languages have been proposed. There are two different viewpoints in the area of Web service composition. Web service Orchestration combines available services by adding a central coordinator (the orchestrator) that is responsible for invoking and combining the single sub-activities. An example is BPEL (Business Process Execution Language). Web service choreography does not assume the exploitation of a central coordinator but rather defines complex tasks via the definition of the conversation that should be undertaken by each participant. Following this approach, the overall activity is achieved as the composition of peer-to-peer interactions among the collaborating services. An example is WS-CDL (short for Web Service Choreography Description Language).

1.1 JAX–RPC

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web Services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems. In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer.

On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy. With JAX-RPC, clients and Web Services have a big advantage, the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web Service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.

The main contribution of this paper will be analyzing the techniques available for verification of the composed web services can be summarized as follows:

- We have taken a survey report for verification of composed web services Called Buchi Automata including algorithm called Dead Marking Algorithm. It is suitable for both deterministic and non-deterministic system.
- In Dead Marking, there is no enable binding element. The final state of service process instance is one of dead marking. If the number of dead
markings reported by state space analysis is more than expected, then there must be an error in the composition of web services or service process design.

- We report our system is giving efficient result than other system by means of accepting regular languages.
- We show that our system is readily applicable for existing applications, and that it can be used for verifying composed web services.

The rest of this paper is organized as follows. In Section 2, describes a related work. In Section 3, we discussed about the Transition Diagram of the proposed work. In Section 4 describes Proposed Work? In Section 5, we discussed the Experiment and Results of the algorithm and performance shown. In Section 6 depicts the conclusion of the work and give an outlook on future research.

2 Related Works

Automata are a mathematical model represents the behavior of the systems. After composing the web services, only limited number of verification techniques is available. Even though the techniques are used, it is applicable only for deterministic system and not applicable for non-deterministic system.

We now discuss related work on Web Service Composition Verification. Recent survey [1] which can be roughly divided into (i) approaches that are based on Timed Automata [1, 4, 10, 11 and 12]. These papers based on Timed Automata and Computation Orchestration. WSCDL used for composition of Web Services, converted into Timed Automata and verified using the tool called UPPAAL for verification of the web service composition.

Workflow net models are used and define a fully-automatic translation of this formalism into conceptual BPEL by means of the tools 4BPEL framework. The automatic generation of BPEL code by using the workflow net formalism as design model, which can be checked of being free of deadlocks and live locks.

This proposal then takes the orchestration viewpoint, whereas, we take choreographic one. Timed Automata applicable only for Deterministic system rather than non-Deterministic System.

We discuss the (ii) Approaches that are based on Interface Automata [2, 8 and 13]. It is based on the verification of the composed web services using interface automata. Initially composition done BPEL4WS, converted into Interface automata mapped into Promela and then verified using the tool called SPIN. This approach is applicable for Deterministic system rather than Non-Deterministic system.

Presented a universal proposal for describing choreographies and orchestrations of Web Services using this formalism. However, only an informal mapping between WS-BPEL orchestrations and process algebra is presented, and as the authors recognize, further work is required. This approach is applicable only for Deterministic System rather than Non-Deterministic System.

We now discuss (iii) Approaches that are based on Petri net [3, 5, 6, 7, 14, 15, 16 and 17] used Petri-nets presented a methodology for the design, verification and validation of composite Web Services using WS-CDL as the language for describing
Web Service interactions, and Petri nets as a formalism (that allows us to analyze the described systems). In this work they have considered timed automata and prioritized collaboration in composite Web Services, so the considered model of Petri nets is a prioritized version of Timed Petri nets. Different Web Services choreography description language (WSCI), which was one of the first proposals for choreography descriptions. However, WS-CDL has a more affluent expressivity than WSCI, as stated in the comparative work of [9].

Verifying Web Services Composition based on LTL and Colored Petri Net [18]. They used the tool combining CPN with LTL to check the correctness of the web services composition. The properties like Reachability and safety are reduced, but it is not concentrating the emptiness problem and dead marking. But Priorities are not considered in this paper.

An inadequate number of verification models are available for the composed web services, even though the methods are available; it is pertinent for deterministic system only and not apposite for non-deterministic system. Moreover the models are at design level only and not present in the implementation level. The models are not satisfying the properties like Safety, Deadlock, Correctness, Dead Marking, only few parameters satisfied by few models in the above mentioned models. Because the above mentioned models are suitable for deterministic system rather than non-deterministic system.

3 Proposed Work

3.1 Overview

Web services choreography is one of the latest techniques for composing Web services. A few works have been carried out in the past using choreography but they fail to address the problems like Correctness, Dead Marking etc., and this motivated to propose an algorithm using WS-CDL. The main objective of this work is to propose a new algorithm for verifying Web services composition using Buchi automata. WS-CDL is used for composing Web services and then WS-CDL is mapped into Buchi automata. This automaton generates the Transition Diagram and the correctness properties of Web services composition can be verified using the algorithm.

3.2 Web Services Choreography

Here WS-CDL is used for composing the Web services [1]. The problem that concerned here is the first step of this scenario – given a request $r$, finding right Web services for $r$. In particular, in this case where one has to combine multiple Web services to satisfy $r$ since no single one can. Consider the following motivating example
Example
Suppose there are two Web services available in the registry. (1) GetPlacesService returns the name of the tourist places according to the given input district. (2) ReserveTicket returns the confirmation for the ticket reserved to the particular tourist places. I want to reserve the ticket for the particular place in a given district. Let us call this request as r. Note that neither of two Web services can satisfy r alone. GetPlacesService can only provide the name of the tourist places and reserveTicket can only for reserving the ticket to the particular tourist places. Thus there is a need to combine both the services to satisfy the request r. Here WS-CDL is used for composing the Web services.

The purpose of the Web Services Choreography Description Language (WS-CDL) is to define multi-party contracts, which describe the externally observable behavior of Web services and their clients (usually other Web services), by describing the message exchanges between them.

3.3 Web Services Verification

Web service is composed by using WS-CDL. For verification WS-CDL is converted into Transition Diagram. Using this transition a new algorithm called DM (Dead Marking algorithm) can be verified. In verification process the following property is verified [4]. It is defined as

Dead marking: Markings having no enabled binding element. The final state of process instance is one of dead marking. If the number dead markings reported by state space analysis tool is more than expected, then there must be errors in the design.

4 Experiment and Results

In this section, we describe a typical scenario in the tourist domain to motivate our work. This scenario will also be used as a running example to illustrate the key concepts throughout the paper. Consider an industrialist, Karna who is running a travel agency that provides comprehensive travel packages to its users. Karna exploits the state of the art Web service technologies to improve the competitiveness of this travel agency. A variety of travel related services (e.g., hotel, tourist places and transport) are involved to satisfy different user requirements. To satisfy the different user requirements the related services are composed.

The tourist scenario contains the services like Hotel Service, Places Service and Transport Service.

Hotel Service lists out the Hotel information in a given district. Also, it provides the services for checking the room availability and reserving the room. In HotelService, getHotels method is used to determine the hotel information; checkRoomavailability method is used to find the availability of the rooms.

Places Service provides the details about the tourist places based on the request of the user. In Places Service, getPlaces method is used to determine the tourist places. For executing this method, district name is passed as an argument.
Transport Service provides the functionality such as checking the seat availability, reserving the ticket, cancel the ticket. In TransportService, checkAvailability method is used to determine the availability of the seats in a given travel on a particular date. reserveTicket method is used to reserve the ticket to a particular place on a particular date. The web services are created by using JAX-RPC and composed by using WS-CDL (Web Services Choreography Description Language). This WS-CDL can be mapped into Buchi Automata. This automaton generates the State Transition Diagram. Using the algorithm this diagram checks the Dead Marking problems.

Fig1: Composed Web Services

4.1 Buchi Automata Generation: The sample code

The WS-CDL is converted into BA for the verification purpose. The following is the sample code for the BA generation.

```java
public String generateba() throws ParserConfigurationException, SAXException, IOException, XPathExpressionException {
    String head = "<?xml version="1.0" encoding="UTF-8" ?>";
    String bas = "<ba>";
    String bac = "</ba>";
    String declaration = "<declaration>urgent chan urgent_chan;int [0,1] ";
    String tname = "<name>System</name>";
    String declarationc = "</declaration>";
    String location = "<location id=";String locationc = "</location>";
    String template = "<template>";String templatec = "</template>";
    String name = "<name>";String namec = "</name>";
    String initchoice = "Init_Choice";String endworkunit = "End_Workunit";
    String innter = "Init_Interaction";String endinter = "End_Interaction";
    String initref = "<init ref=";int inter = 0;int initcho = 0;int endwork = 0;}
```

4.2 State Transition Diagram

Each service can be represented as a state. The changes between the states with the help of input are said to be transition. The following diagram shows the transition diagram for the composed web services represented in the Figure 1. This figure shows the composed web services using the language called WSCDL and figure 2 generates the transition diagram for composed web services ie fig 1.
The above mentioned verification parameters are verified by using the DM algorithm.

### 4.3 Verification Algorithm

The above algorithm identifies the dead marking states. So that we can easily identify whether our design is correct or not. The algorithm initially create machine description which includes the total number of states, initial state, transition, final state and input. Verification can be performed by using two parameters, one is Directory and machine description. The directory contains the marked dead state information. If the Particular input reaches to the specified state indicates that there is no dead transition between the states. If the state is $\emptyset$ represents dead mark is true, will be maintained in the directory. It indicates that the design of composed service is badly designed. Again we have to recompose the web services. This algorithm is functioning in the above transition diagram ie Fig 2.

### 4.4 Performance

The Dead Marking problem is solved by the verification of the composed services. When giving the input, it should reach the final state from the starting state. If it does not reach the final state, automatic recomposition will be done and again it will verify
the dead marking against the accepting state (final state). If this property is verified successfully then the safety and liveness properties will be automatically verified.

### 4.4.1 Formula

\[ \text{Dead Marking Factor} = \frac{\text{Number of final states moved}}{\text{Total number of states}} \]

### 4.4.2 Verification models

**Table 1**: comparing existing system performance

<table>
<thead>
<tr>
<th>Dead Marking</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timed Automata</td>
<td>0%</td>
</tr>
<tr>
<td>Interface Automata</td>
<td>0%</td>
</tr>
<tr>
<td>Colored Petrinet</td>
<td>50%</td>
</tr>
<tr>
<td>Orchestration via computation of timed Automata</td>
<td>0%</td>
</tr>
<tr>
<td>Refinement checking</td>
<td>0%</td>
</tr>
<tr>
<td>Buchi Automata</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Graph 1**: Performance Comparison

Table 1 shows the performance report of the existing system with proposed work in terms of Dead Marking, the models are Timed Automata, Interface Automata, colored Petrinet, Orchestration via computation of timed Automata, Refinement checking are not measuring the above mentioned parameter. But Colored Petrinet model giving the 50% of performance. This model also giving less performance than my proposed work (Buchi Automata 100%). Table 1 is explained in the table 2.

Graph 1 shows the pictorial representation of the existing system with proposed work.

**Table 2**: Comparing existing system with proposed system

<table>
<thead>
<tr>
<th>S.No</th>
<th>Researcher</th>
<th>Formal Model</th>
<th>Transformation Verification</th>
<th>Type of System</th>
<th>Specification</th>
<th>Tool Used</th>
<th>Level of Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>J. Seng Deng</td>
<td>Timed Automata</td>
<td>Yes</td>
<td>Deterministic System</td>
<td>WSCEL</td>
<td>WS and UPFAL</td>
<td>Design Level</td>
</tr>
<tr>
<td>2</td>
<td>J. Seng Deng</td>
<td>Interface Automata</td>
<td>No</td>
<td>Deterministic System</td>
<td>BPEL</td>
<td>Parasol</td>
<td>Implementation Level</td>
</tr>
<tr>
<td>3</td>
<td>J. Seng Deng</td>
<td>Orchestration via computation of timed Automata</td>
<td>No</td>
<td>Deterministic System</td>
<td>LTL</td>
<td>UPFAL</td>
<td>Design Level</td>
</tr>
<tr>
<td>4</td>
<td>J. Seng Deng</td>
<td>Colored Petrinet</td>
<td>Yes</td>
<td>Partial</td>
<td>BPEL</td>
<td>CPU</td>
<td>Implementation Level</td>
</tr>
<tr>
<td>5</td>
<td>J. Seng Deng</td>
<td>Refinement Checking Method</td>
<td>Yes</td>
<td>No</td>
<td>BPEL</td>
<td>UPFAL</td>
<td>Design Level</td>
</tr>
<tr>
<td>6</td>
<td>Our Work</td>
<td>Buchi Automata</td>
<td>Yes</td>
<td>Both Deterministic and non-Deterministic System</td>
<td>WSCEL</td>
<td>WS and CVT</td>
<td>Implementation Level</td>
</tr>
</tbody>
</table>

Table 2 shows the detailed performance report of the existing and proposed model.
5 Conclusion

The Internet is going through several major changes. It has become a vehicle of Web services rather than just a repository of information. Web Services are the most famous implementation of service-oriented architectures that has brought some challenging research issues. One of these is the composition

- Combining functionality provided by other web services is the Composite service.
- Process of developing composite web services is the service composition.
- Create new value added services by incorporating some existing web services together.

Even though the verification techniques are available for verifying the composed Web Services, those techniques are not suitable for non-deterministic system. Those methods not covering the Dead marking problems. So a new algorithm for the verification of Web services choreography was presented. The Web services have been created and the corresponding wscdl for the Web services is generated. In Web service choreography the interaction between the services are specified in WS-CDL. Then, we present a technique for the translation of WS-CDL into Transition Diagram through Buchi Automata. A new algorithm called Dead Marking algorithm was developed. It checks whether our composed service correct or bad designed.

References