Tools for Automated Verification of Web services

Tevfik Bultan, Xiang Fu, and Jianwen Su

Department of Computer Science
University of California
Santa Barbara, CA 93106, USA
{bultan,fuxiang,su}@cs.ucsb.edu

1 Introduction

Web-based software applications which enable user interaction through web browsers have been extremely successful. Nowadays one can look for and buy almost anything online using such applications, from a book to a car. A promising extension to this framework is the area of web services, web-accessible software applications which interact with each other using the Web. Web services have the potential to have a big impact on business-to-business applications similar to the impact interactive web software had on business-to-consumer applications.

There are various issues that have to be address in order to develop successful web services: a) services implemented using different platforms (such as .NET or J2EE) should be able to interact with each other; b) it should be possible to modify an existing service without modifying other services that interact with it; c) services should be able to tolerate pauses in availability of other services and slow data transmission. Web services address these challenges by the following common characteristics: 1) standardizing data transmission via XML [13], 2) loosely coupling interacting services through standardized interfaces, and 3) supporting asynchronous communication. Through the use of these technologies, Web services provide a framework for decoupling the interfaces of Web accessible applications from their implementations, making it possible for the underlying applications to interoperate and integrate into larger, composite services.

A fundamental question in developing reliable web services is the modeling and analysis of their interactions. In the last couple of years, we developed a formal model for interactions of composite web services, developed techniques for analysis of such interactions, and built a tool implementing these techniques [3–8]. Below we give a brief summary of these contributions.

2 Conversations

Our work focuses on composite web services which interact with asynchronous messages. We call each individual web service a peer. A composite web service consists of a set of peers which interact with each other using asynchronous messages [3]. Such a system can be modeled as a set of state machines which communicate using unbounded FIFO message queues, as in the communicating

finite state machine model [2]. When a message is sent, it is inserted to the end of the receiver's message queue. In our model, we assume that each peer has a single queue for incoming messages and receives the messages in the order they are inserted to the queue.

In order to analyze interactions of asynchronously communicating web services we first need a formal model for their interactions. We model the interactions in such a system as a conversation, the global sequence of messages that are exchanged among the web services [3, 9, 11]. Note that, a conversation does not specify when the receive events occur, it only specifies the global ordering of the send events.

Given a composite web service, one interesting problem is to check if its conversations satisfy an LTL property. Due to asynchronous communication via unbounded FIFO queues this problem is undecidable [4].

3 Realizability and Synchronizability

A composite web service can be specified in either top-down or bottom-up fashion. In the top-down approach the desired conversation set of the composite web service is specified as a conversation protocol. In the bottom-up approach each peer is specified individually and a composite web service is specified by combining a set of asynchronously communicating peers. For both approaches, our goal is to verify LTL properties of the set of conversations generated by the composition.

There are two interesting properties within this framework: realizability and synchronizability. A conversation protocol is realizable if the corresponding conversation set can be generated by a set of asynchronously communicating web services. On the other hand, a set of asynchronously communicating web services are synchronizable if their conversation set does not change when asynchronous communication is replaced with synchronous communication. We developed sufficient conditions for realizability and synchronizability that can be checked automatically [4, 5, 7].

Using the realizability analysis, reliable web services can be developed in a top-down fashion as follows: 1) A conversation protocol is specified and checked for realizability; 2) The properties of the conversation protocol are verified using model checking; 3) The peer implementations are synthesized from the conversation protocol via projection.

Similarly, synchronizability analysis enables development of reliable web services in a bottom-up fashion. If a web service composition is synchronizable, we can verify its behavior without any input queues and the verification results will hold for the asynchronous communication semantics (with unbounded queues).

4 Web Services Analysis Tool

We developed a tool which implements the techniques mentioned above. Web Service Analysis Tool (WSAT) [8, 12] verifies LTL properties of conversations

and checks sufficient conditions for realizability and synchronizability. In order to model XML data, WSAT uses a guarded automata model where the guards of the transitions are written as XPath [14] expressions. This guarded automata model provides a convenient intermediate representation. The front end of the WSAT translates web services specified in BPEL [1] to this intermediate representation [5]. WSAT uses the explicit-state model checker SPIN [10] for LTL model checking by translating the guarded automata model to Promela [6]. In the future, we plan to investigate symbolic analysis and verification of web services.

Acknowledgments. Authors are supported by NSF Career award CCR-9984822, and NSF grants CCR-0341365, IIS-0101134, and IIS-9817432.

References

1. Business Process Execution Language for Web Services (BPEL4WS), version 1.1. http://www.ibm.com/developerworks/library/ws-bpel.
2. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.
3. T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: A new approach to design and analysis of e-service composition. In *Proc. of the 12th Int. World Wide Web Conference (WWW 2003)*, pages 403–410, May 2003.
4. X. Fu, T. Bultan, and J. Su. Conversation protocols: A formalism for specification and verification of reactive electronic services. In *Proc. of 8th Int. Conf. on Implementation and Application of Automata (CIAA 2003)*, volume 2759 of *LNCS*, pages 188–200, 2003.
5. X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL Web Services. In *Proc. of the 13th Int. World Wide Web Conf. (WWW 2004)*, pages 621–630, May 2004.
6. X. Fu, T. Bultan, and J. Su. Model checking XML manipulating software. In *Proc. of the 2004 ACM SIGSOFT Int. Symp. on Software Testing and Analysis (ISSTA 2004)*, pages 252–262, July 2004.
7. X. Fu, T. Bultan, and J. Su. Realizability of conversation protocols with message contents. In *Proc. of the 2004 IEEE Int. Conf. on Web Services (ICWS 2004)*, pages 96–103, July 2004.
8. X. Fu, T. Bultan, and J. Su. WSAT: A tool for formal analysis of web service compositions. In *Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV 2004)*, pages 510–514, July 2004.
9. J. E. Hanson, P. Nandi, and S. Kumaran. Conversation support for business process integration. In *Proc. of 6th IEEE Int. Enterprise Distributed Object Computing Conference*, 2002.
10. G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, Boston, Massachusetts, 2003.
11. IBM. Conversation support project. http://www.research.ibm.com/convsupport/.
12. Web Service Analysis Tool (WSAT). http://www.cs.ucsb.edu/ su/WSAT.
13. Extensible markup language (XML). http://www.w3c.org/XML.
14. XML Path Language. http://www.w3.org/TR/xpath.