Formal Framework for Automated Analysis and Verification of Web-based Applications

May Haydar
Département d'informatique et de recherche opérationnelle, Université de Montréal
CP 6128 succ. Centre-Ville, Montréal, Québec, H3C 3J7, Canada
mhaydar@crim.ca

Abstract

We present an ongoing Ph.D. research developing a formal approach for modeling an existing web application using communicating finite automata model. We build the automata from a recorded browsing session. The obtained model could then be used to verify user-defined properties of the application with a model checker. We present an implementation of the approach that uses the model checker Spin.

1. Motivation

The steep growth of the production of web applications (WA) and the numerous constantly evolving technologies used in the development of such applications have led to an increased complexity of maintaining high quality for WAs. In particular, developers do not have sufficient tool support to create high quality applications. Thus development tools should be complemented with analysis and validation tools and methods. Numerous web testing tools, which target this concern, already exist. They generally verify the syntax of HTML documents, confirm the hyperlink integrity of a set of HTML documents, test the GUI components embedded in the browsers, and measure the performance of the WA. In [20], over 250 links to such commercial tools are provided.

In recent years, the software community started to acquire formal methods as a practical and reliable solution to analyze applications in various domains. The literature shows how formal methods are being used in the white-box and black-box analysis of applications in various domains. In particular, methods and tools that consider the analysis of existing systems based on their behaviors (dynamic black box approach) have become quite common. This constitutes the case for WAs, where access to English specifications and lengthy design documents is not easy, if they existed. In this work, we plan to set a formal framework, which allows us to analyze existing WA and to implement proper tools to automate the analysis process.

2. Problem Statement

The proposed research addresses the following main problems:

1. Inferring automata based models from a web application under test (WAUT), independently from the browser, and without any access to the underlying code, to verify user-defined properties.
2. Defining property patterns and anti-patterns related to good and bad practices in developing WAs, to alleviate the burden on the user/tester to learn temporal logic foundations to specify properties.

One could argue that thorough analysis could be performed on the actual code of the WAUT. However, such approach reduces the application domain eliminating the possibility of analysis of a wide range of WAs where the code is not available. We rely on a non-intrusive dynamic approach by inferring automata models from the run-time behavior of the WAUT. On the other hand, one could build specific tools [12,15] for verifying a restricted range of properties, usually predefined and embedded in the tools. Such tools do not allow the developer to verify a wide range of properties on a WAUT model, and do not scale to relatively large models. Hence, using formal methods, we could benefit from the availability of various reliable tools, used for several years in industry and academia. Such tools allow the specification of general properties using temporal logic, thus, solving a wider range of problems related to WAUT. Also, these tools have undergone years of development, enhancement, and upgrades to solve many of the scalability problems related to the state explosion problem [8].

3. Research Objectives

The main objective of the proposed research is to develop an integrated framework and a prototype tool environment to automate verification of properties of WAs. Developing a prototype verification environment for WAs, we use an off-the-shelf model checker, Spin [9]. The main research goals are as follows:

1. Devise methods to analyze intercepted HTTP transactions and to extract automata based models, which are fed to a model checker to verify properties of interest. These methods should be adequate for modeling WAs that exhibit concurrent behavior when developed using frames and multiple windows, and for modeling both stateless WA and stateful WA (cookies, databases).
2. Define property patterns and anti-patterns, and provide a library that could be used by the web tester and alleviate the task of learning a temporal logic and formulating properties.

4. Related Work

Formal modeling of web applications is a relatively new research direction. Previous work on the topic includes modeling approaches that target the verification of such applications [11,12,13,14], testing [16,17,18,19], and design and implementation [2,3,4,5,6]. In [11,12] an approach is presented where a web site is modeled as a directed graph. A node in the graph represents a web page and the edges represent links clicked. If the page contains frames the graph node is then a tree whose tree nodes are pages loaded in frames and tree edges are labeled by frame names. This model is used to verify properties of web sites with frames. However, only static pages are considered in this work, concurrent behavior of multiple windows is not modeled, and all the links whose targets could create new independent windows are treated as broken links. Besides, any frameset that could be present in the application is completely ignored. Also, in the model, a page loaded in an unnamed window (as a result of the predefined target "_blank" associated with a link) is represented as a graph node that replaces the existing node as if the page is loaded on top of page where the link was clicked; this incorrectness is due to the inadequacy of the proposed model to represent concurrent behavior of multiple windows. In [13,14] a model based on Petri nets to model check static hyperdocuments [13] and framed pages [14] is presented. While Petri nets can express parallel and concurrent behavior, the authors build the overall state space as input of the model checker, which is a tedious and erroneous approach especially with large applications with several frames and windows. [11,12,13,14] do not tackle the modeling and verification of form-based pages that are dynamically generated by a server program, neither concurrent behavior of applications with multiple windows. The work in [16,17] focuses on inferring a UML model of WAs. This model, merely a class diagram, is mainly used for the static analysis of WAs: HTML code inspection and scanning, data flow analysis, and semi automatic test case generation. In [18], the above mentioned modeling technique is extended such that a WA is executed to extract models for dynamic web pages using server's access logs. These logs present limited information on the requests since only the request headers are logged. In case dynamic pages are generated based on POST method requests, the form data submitted is usually stored in the message body of the request; thus, making those pages requests undistinguishable and introduce unnecessary non-determinism into the resulting model. Besides, the approach is inadequate for modeling concurrent behavior of frames and multiple windows. In [19], a modeling technique for Was is presented based on regular expressions. The focus is on modeling the behavior of WAs, consisting of merely dynamically generated pages, for the purpose of functional testing. Other approaches for modeling WAs are oriented towards the design rather than analysis of WA. These include object oriented based models [2] and statechart based models [3,4,5,6], that are tailored to forward engineering, logical and hierarchical representation of WAs. Such models are not available for analyzing existing WA developed without formal models. Each of the existing related work concentrates on some aspects of WAs leaving out other aspects that remain untouched, or unfeasible to model using the corresponding proposed approach. These attempts indicate that formal modeling of WA is still an open complex problem especially when it comes down to modeling multiple frames and windows, and properties which have to reflect various concerns of different stakeholders of WAs.

5. Our Approach

In this work, we attempt to develop a modeling approach that could produce a finite automaton model tuned to features of WA that have to be validated, while delegating the task of property verification to an existing model checker. We elaborate a black-box (dynamic) approach by executing the WAUT and analyzing only its external behavior without any access to server programs or databases. The observations are provided by a network monitoring tool [1,7], where

HTTP requests and responses are logged. Our model is a system of communicating automata representing all windows, frames, and framesets of the application under test. The existence of frames, framesets, and windows reflects concurrent behavior of the WAUT, where these objects affect each other behaviors via links and forms with specified targets. Therefore, a suitable and natural modeling technique is communicating automata, so that a global state graph of the model can be constructed by a model checker verifying a given property. As opposed to the existing approaches, we model not only static pages, but also dynamic pages with form, frames and frameset behavior, multiple windows, and their concurrent behavior. Generally speaking, one could build a special web-oriented model checker, as in [12], to verify specific properties. This task requires the building of all the necessary algorithms from scratch. We opt to the use of an existing model checker, Spin [9], used in several industrial applications, such that we only have to describe our model in Spin's input language.

6. Research Results

We currently consider stateless WA that relies on the stateless HTTP protocol where an external behavior of the application consists of requests and responses. Current results can be found in [21]. A behavior of a WAUT, we call it a *browsing session*, is interpreted as a sequence of web pages that have the same domain name intermittent with the corresponding requests.

6.1. Browsing Sessions

An observed browsing session represents the behavior of the communicating entities of the WA, namely, browser's main and independent windows, frames and framesets. Therefore, given the browsing session, we first determine *local browsing sessions* that correspond to the behaviors of the entities in the browsed part of the WA. These entities affect each other's behavior through target names associated with links and forms. We assume frames, framesets, and windows are initially in an inactive state. They are activated (created) through link clicks or form submissions with target names in case of windows, or frame source URIs in case of frames. Frames/framesets are deactivated (destroyed) whenever a link/form is clicked/submitted and target a parent of these frames of framesets. Whenever an explored link/form targets an active entity, the entity where the link/form was triggered does not change its display, while the targeted entity changes its display.

6.2. Automata Model of a Browsing Session

An observed browsing session is modeled by a system of communicating automata, such that each window, frame, and frameset is represented by an automaton. This is achieved by converting each of the local browsing sessions into an automaton as follows. Initially, the automaton is in the start state, which is inactive state. All the response pages in the local session are converted into states of the automaton. Page attributes that are of interest to the user represent the state attributes, or atomic propositions, that are true in the states and used for model checking. The set of events include the set of all the links, form actions, frame source URIs present in all the pages displayed in the corresponding entity, as well as all the requests in the local session. Each request in the local session defines a transition from the state/page where the request was triggered to the state/page that is the response of the request. Each explored form or link in a page loaded in the entity and that is repeated in another page defines a transition from the state where it occurs to the state that corresponds to the response of the submitted filled form or clicked link. Each unexplored link or unexplored form defines a transition from the state representing the page where it exists to a state called *trap* that represents the unexplored part of the WA and whose attributes are undefined.

Communications between automata occur in the following three cases. (1) Whenever a request for a link/form in an entity targets another entity. (2) Whenever frames are created (displayed in a window). (3) Whenever a link/form in a frame/frameset targets a parent entity. In [21], we presented the algorithms that convert a browsing session into communicating automata and how the communication between entities is represented as rendezvous.

6.3. Implementation of the Approach

Our approach is implemented following the framework illustrated in Figure 1. The user defines some desired attributes through a graphical user interface prior to the analysis process. These attributes are used in formulating the properties to verify on the application. A monitoring tool intercepts HTTP requests and responses during the navigation of the WAUT. The intercepted data is fed to an analysis tool that continuously analyzes the data in real time (online mode), incrementally builds an internal data structure of the automata model of the browsing session, and translates it into XML-Promela. The XML-Promela file is then imported into aSpin [10], an extension of Spin model checker [9]. aSpin then verifies the model against the properties and the model checking results include counter-examples that facilitate error tracking. Preliminary experiments show promising results.

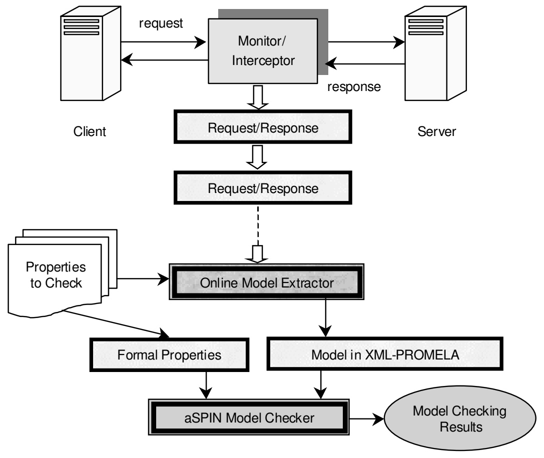

Figure 1. Framework of the Approach

7. Future Research

Our current methods are based on the assumption that we observe stateless HTTP based behavior of WA. As a future extension, we can model stateful WA behavior which is based on client side (cookies) and server side (databases) state. Also, our framework allows the verification of properties that could be related to several features of WAs. These include reachability properties, deadlocks and livelocks, frames behavior related properties, and their mixture, as well as various intricate and thus difficult to detect properties related to the concurrent behavior of the different entities of the WA. Properties can also be attribute-related where the user specifies some desired features based on page attributes. Therefore, we intend to analyze and formalize what is known to be good and bad practices in the development of WA. This analysis will contribute to the building of a library of property patterns and anti-patterns related to WA that helps the user/tester easily specify properties without having to learn a temporal logic and master the tricky process of property specification.

8. References

[1] A. Luotonen, *Web Proxy Servers*, Prentice Hall, 1998.

[2] J. Conallen, "Modeling Web Application Architectures, with UML", *In Proc. of Communications of the ACM*, October 1999, vol. 2, No. 10.

[3] M.C.F. de Oliveira, P.C. Masiero, "A Statechart-Based Model for Hypermedia Applications", *ACM Transactions on Information Systems*, Vol. 19, No. 1, 28-52, January 2001.

[4] F.B. Paulo, P.C. Masiero, M.C.F. de Olivieira, "Hypercharts: Extended Statecharts to Support Hypermedia Specification", *IEEE Transactions on Software Engineering*, Vol. 25, No. 1, Jan. 1999.

[5] F.B. Paulo, M.A.S. Turine, M.C.F. de Olivieira, "XHMBS: a Formal Model to Support Hypermedia Specification", *In Proc. of the 9th ACM Conference on Hypertext*, United Kingdom, June 1998.

[6] K.R.P.H. Leung, L.C.K. Hui, S.M. Yui, R.W.M. Tang, "Modeling Web Navigation by Statechart", *In Proc. of the 24th IEEE Annual International Computer Software and Applications Conference*, Taiwan, October 2000.

[7] "Ethereal, Network Protocol Analyzer", http://www.ethereal.com/.

[8] E. M. Clarke, O. Grumberg, D. A. Peled, *Model Checking*, MIT Press, 2000.

[9] G.J. Holzmann, *The Spin Model Checker, Primer and Reference Manual*, Addison-Wesley, 2003.

[10] "aSpin Model Checker", http://polaris.lcc.uma.es/~gisum/fmse/tools/mainframe.html.

[11] L. de Alfaro, "Model Checking the World Wide Web", *In Proc. of the 13th International Conference on Computer Aided Verification*, Paris, France, July 2001.

[12] L. de Alfaro, T.A. Henziger, F.Y.C. Mang, "MCWEB: A Model-Checking Tool for Web Site Debugging", *Poster, 10th WWW Conference*, Hong Kong, 2001.

[13] P.D. Stotts, C.R. Cabarrus, "Hyperdocuments as Automata: Verification of Trace-Based Browsing Properties by Model Checking", *ACM Transactions on Information Systems*, Vol.16, No. 1, Jan. 1998, 1-30.

[14] P.D. Stotts, J. Navon, "Model Checking CobWeb Protocols for Verification of HTML Frames Behavior", *In Proc. of the 11th WWW Conference*, Hawai, U.S.A., May 2002.

[15] M. Benedikt, J. Freire, P. Godefroid, "VeriWeb: Automatically Testing Dynamic Web Sites", *In Proc. of the 11th International World Wide Web Conference*, Hawai, U.S.A., May 2002.

[16] F. Ricca, P. Tonella, "Web Site Analysis: Structure and Evolution", *In Proc. of International Conference on Software Maintenance (ICSM'2000)*, pp. 76-86, San Jose, California, USA, October 11-14, 2000.

[17] F. Ricca and P. Tonella, "Analysis and Testing of Web Applications", *In Proc. of the International Conference on Software Engineering (ICSE'2001)*, pp. 25-34, Toronto, Ontario, Canada, May 12-19, 2001.

[18] P. Tonella and F. Ricca, "Dynamic Model Extraction and Statistical Analysis of Web Applications", *In Proc. of International Workshop on Web Site Evolution (WSE 2002)*, pp. 43-52, Montreal, Canada, October 2, 2002.

[19] Y. Wu, J. Offutt, "Modeling and Testing Web-based Applications", GMU ISE Technical ISE-TR-02-08, November 2002.

[20] "Web Site Test Tools and Site Management Tools", http://www.softwareqatest.com/qatweb1.html.

[21] M. Haydar, A. Petrenko, H. Sahraoui, "Formal Verification of Web Applications Modeled by Communicating Automata", *In Proc. of 24th IFIP International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2004),* Madrid, Spain, September 2004.