An Approach to Test Generation for Web Applications

Pan Liu[1,3], Huaikou Miao[2], Hongwei Zeng[2] and Lizhi Cai[3]

[1]*College of Computer Engineering and Science, Shanghai Business School, Shanghai 201400, China*
[2]*School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China*
[3]*Shanghai Key Laboratory of Computer Software Testing & Evaluating, Shanghai, China*
panl008@163.com

Abstract

The paper presents a method of test generation for Web applications based on model partition. Compared with the method of test tree for test generation, our approach ensures the effectiveness of test paths generated from the model of Web applications. In this paper, we first employ the Kripke structure to modeling the navigation behavior of a Web application. Then the theory for model partition is proposed to solve the problem of inefficiency test paths. Next a frame of test generation based on model partition is designed to support test generation and redundancy reduction of test paths. Finally, we take the student course system as an example to illustrate our approach.

Keywords: *test generation; model partition theory; effective test paths; the frame of test generation; Web application*

1. Introduction

Web applications currently make up one of the largest growth areas in software [1]. However, the inherent complexity of web applications brings the new challenge for traditional test methods. It is difficult to discover where web applications are failing and why they are failing.

Traditionally, the QA team tests web applications by manual testing. When web applications become more complex, this way is not feasible for two reasons. Firstly, for the tester, it is hard to remember what action sequences were taken to reproduce the error as a bug is encountered. In addition, manual testing of WAs is tedious and time consuming, especially in regression testing, a small portion of the entire web application is tested repeatedly.

Therefore, automatic test generation becomes an importation field of research for conquering the shortcomings of manual testing. Model-based testing is an effective automated generation technique for WAs, including FSM-based test generation [1, 2], UML model [3] and TestUML method [4], User session data method [5] and State-based testing method [6], etc. The leading idea of model-based testing is to use models defined in software construction to drive the testing process, in particular, to automatically generate test cases. The faults in software can be found by observing inconsistency between the software behaviors under

testing and its model (or specification). For that reason the model of the system must be correct and test paths derived from a model of software needs to be corrective and effective.

Test tree, as an assisted method, has been widely employed to generate test paths for web applications in model-based testing. This method needs to traverse the whole model to construct a test tree. Test sequences from root to leaves in the tree are taken as test paths of a Web application. Sometime, test paths derived from test tree are unsafe and inefficient, which will result in failure of model-base testing. For example, for an online forum system, there are two types of users: registered users and guests, where the former is authorized to browse and add the posts on the forum and the latter has only the authority of seeing the posts on the forum. Figure 1 (a) is an abstract page navigation model of the online forum system, and (b) is its test tree. From the test tree, we obtain two test sequences $<s_1, s_2, s_3, s_2>$ and $<s_1, s_2, s_4, s_3>$. However, for the guest, the sequence $<s_1, s_2, s_4, s_3>$ is ineffective because the guest has not been authorized to add the post to the online forum system.

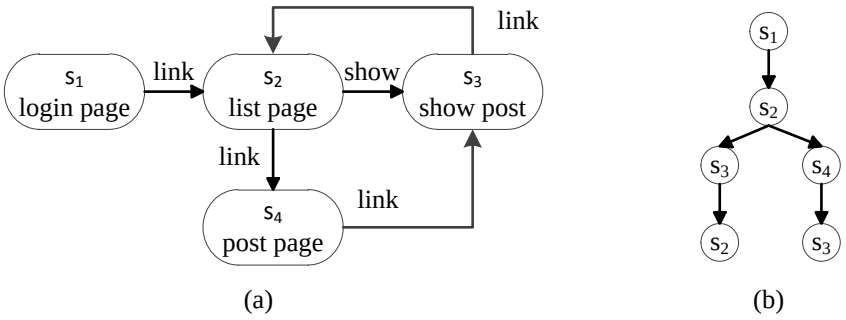


Figure 1. (a) Navigation Model of an Online Forum System and (b) a Test Tree of the Model in (a)

As an automatic, model-based, property-verification approach, model checking can provide a diagnostic sequence (called a counterexample) in the stack as a violation of properties is detected [7, 8]. This technique has been successfully applied to hardware and software verification in the past decades. And some efficient tools for model checkers, such as NuSMV [9] and SPIN [10], have been employed in large real-world applications.

In this paper, we use the method of model checking to decompose the navigation model of Web applications in order to remove the unsafe test paths. Firstly, the Kripke structure is adopted to build a navigation model of a Web application. And then the frame of model partition is presented. According to this frame, we can obtain some navigation sub-models, and the related test trees are constructed from them. Finally, the effective test paths are generated from test trees.

Our contributions in this paper include that (1) we propose the approach to test generation for Web applications based on model partition, and (2) the related theory of model partition is set up in the paper.

The remainder of this paper is organized as follows. Section 2 gives the definition of the navigation model for Web applications. Section 3 introduces the description of the CLT formula of safety property. Theories of model partition are presented in Section 4. Section 5 provides the method for redundancy reduction of test paths based on the specify test intention. The related work is described in Section 6 and Section 7 concludes the whole paper and suggests the direction for our future work.

2. Navigation Model of Web Applications

In this paper, we lay emphasis on testing of WAs' navigation behaviors, including request/response behavior between client and server, and link behavior among the pages. And other properties, such as the calls among software components, the detail content of pages and the inner state of pages, are ignored in the paper. We refer to the Kripke structure [12] to modeling navigation model for Web applications.

Navigation model of a Web application is a classic state transition graph, where node denotes the reachable state (page) and edge denotes the transition of state (page navigation). In pages, we use a set of atomic propositions to mark navigation operations.

Definition 1 (Navigation model). Navigation model of Web applications is a quintuple $PN = (S, Init, AP, R, L)$, where S is a finite set of states denoted by pages in Web applications, $Init \subseteq S$ is a set of initial states, AP is a set of atomic propositions, $R \subseteq S \times S \times AP$ is a transition relation such that for every state $s \in S$ there exist a state $t \in S$ and an atomic proposition $p \in AP$ satisfying $(s, t, p) \in$ R, and $L: S \rightarrow 2^{AP}$ is a state function that labels each state with the set of atomic proposition.

In definition 1, S consists of all pages, such as static pages stored in the server and dynamic pages generated by the outputs according to the user's inputs. We assume the login page of a Web application is taken as the unique initial state. Atomic propositions in AP used to describe the properties of pages, roles and requests. We define atomic propositions "Page-<page name>" as the page, "Role-<role name>" as the role where role's names include T (Teacher role) and S (Student role) in our following illustration, "Link-<page name>" as the link, "Error-<page name>" as the error login, and "Return-<Page name>" as a returning page. Pages and requests in Navigation model of a Web application are alternated. The transition $(s, t, p) \in R$ denotes that the page s triggers a request p and then moves to t, that means t is the response result what Web applications deal with the request p in the page.

Remark: Different from the definition in our previous [8], we take $R \subseteq S \times S \times AP$ as a transition relation and not $R \subseteq S \times S$. If there are two transitions from the state s_1 to the state s_2, (s_1, s_2) cannot distinguish them.

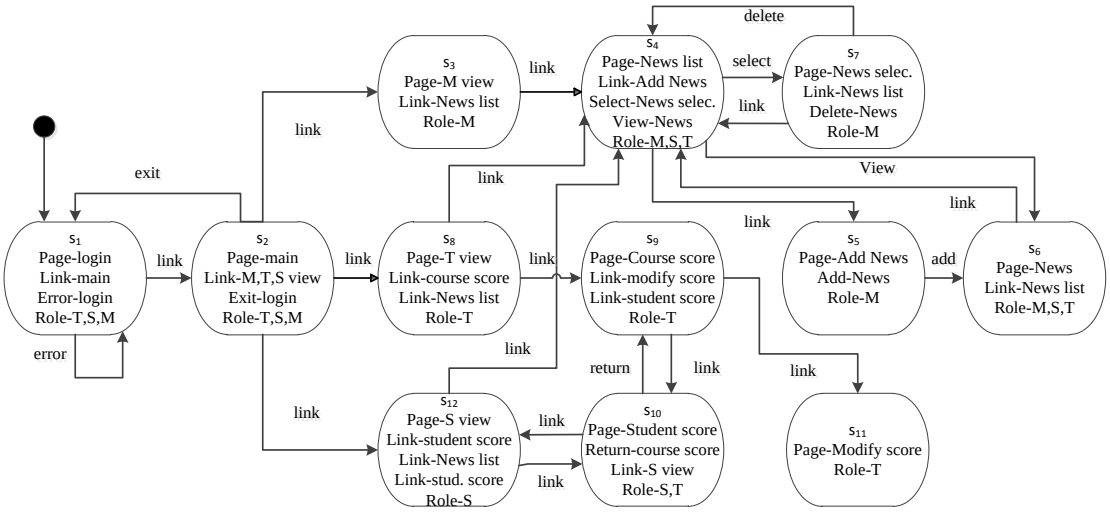


Figure 2. Navigation Model of the Student Course Management System

We take the student course management system with the three user roles, as an illustration, to describe our approach in the paper. According to definition 1, we make the navigation model of this system shown in Figure 2.

In this system, the teacher's role is denoted as Role=T, the student role is denoted as Role=S, and the manager's role is denoted as Role=M. The teacher's role can record and modify the scores of student course, and can check the score of a student. The student role can only see their own course scores. The manager's role can browse, add and delete news. In addition, In addition, Role=M,T,S denotes that the page can be visited by the manager, teacher and student roles, and Role=T,S denotes that the page (state) can be visited by the teacher and student roles.

We construct a test tree shown in Figure 3 (a) by using the breadth-first search way. And then seven root-to-leafs paths taken as test paths are shown in Figure 3 (b). However, test paths <s_1, s_2, s_8, s_4, s_7, s_4> and <s_1, s_2, s_8, s_4, s_5, s_6> are ineffective because the teacher role cannot delete or add news in this system. And test path <s_1,s_2, s_{12}, s_{10}, s_9> is inoperative because the teacher role has not been authorized to return the course scores page for modifying themselves scores. The reason for generating ineffective test paths is that there are a number of unsafe factors in navigation model, which will lead to failure of model-based test method. Therefore, we suggest partitioning navigation model into different sub-models according to the diverse user roles in order to solve these two problems.

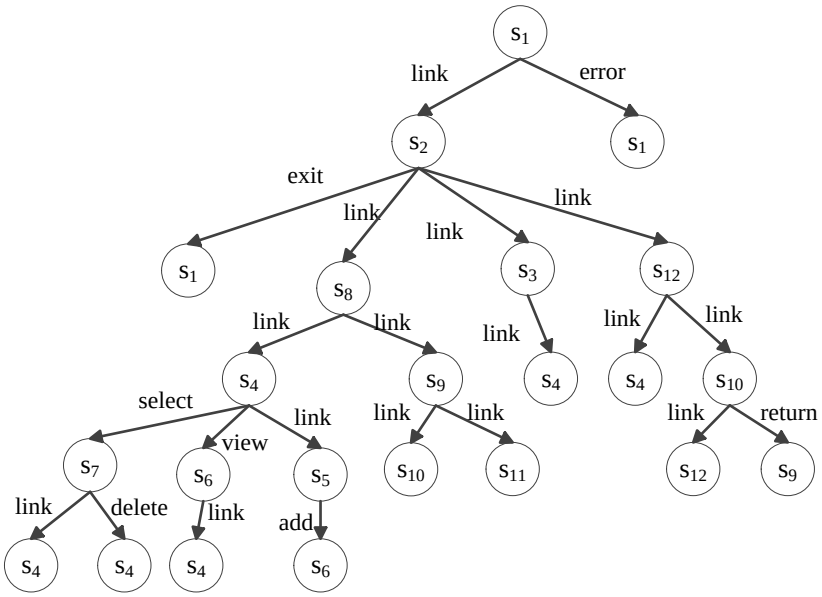


Figure 3. A Test Tree Generated from the Navigation Model in Figure 2

Definition 2 (Effective navigation behavior). Suppose the function F: $U \rightarrow 2^A$, where U ($U \neq \varnothing$) denotes the set of user roles in Web applications and A ($A \neq \varnothing$) denotes the set of role navigation operations. For $\forall u \in U$, $F(u)$ denotes the set of effective navigation behaviors of the user role u.

To obtain the set of uses' effective navigation behaviors, we employ the technique of model checking. Firstly, we adopt Computation Tree Logic (CTL) to describe safety properties of the user role u, and then input both those properties and navigation model into NuSMV to verify them. If counterexamples [12] are outputted in NuSMV, we eliminate all

unsafe behaviors of the user role *u* in navigation model and finally obtain effective navigation sub-models of the user role u.

3. CLT Formula of Safety Property

CTL is a kind of temporal logic, which is used to represent formal properties. We refer to [12] and give the definition of Computation Tree Logic (CTL) as follows.

Definition 3 (Computation Tree Logic, CTL). For $\forall p \in AP$, the syntax of the CTL formulas can be defined via Backus-Naur form:

$$\phi ::= p|\neg\phi \mid \phi \wedge \varphi \mid \phi \vee \varphi \mid \phi \rightarrow \varphi \mid \mathbf{AX}\phi \mid \mathbf{EX}\phi \mid$$
$$\mathbf{AF}\phi \mid \mathbf{EF}\phi \mid \mathbf{AG}\phi \mid \mathbf{EG}\phi \mid \mathbf{A}[\phi \mathbf{U} \varphi] \mid \mathbf{E}[\phi \mathbf{U} \varphi]$$

where p ranges over a set of atomic propositions and ϕ, φ are CTL formulas. Note that $\neg\phi$ is represented as $!\phi$ in the model checker NuSMV.

Each of the CTL temporal connectives is a pair of symbols. The first of the pair is one of A and E. A means ‘along All paths’ (inevitably) from the current state and E means ‘along at (there Exists) one path’ (possibly). The second one of the pair is the temporal operator X, F, G, or U, meaning ‘neXt state’, ‘some Future state’, ‘all future states (Globally)’, and Until, respectively. The paper sets the priority of operators from high to low level: $\neg$, AX, AF, EX, EF and EG; $\wedge$ and $\vee$; $\rightarrow$, AU and EU.

According to the definition of CTL, we give the related safety properties in the student-course management system such that the navigation model can be checked in NuSMV. Safety properties of the student role contain:

- The student has been authorized to browse the news page, but no linking the news page to add news.

$$AG(\text{Role-S} \rightarrow \neg(\text{link} \wedge EF\ \text{Page-add News})) \quad (1)$$

- The student cannot select news in the News list page so that news can be deleted.

$$AG(\text{Role-S} \rightarrow \neg EF\ (\text{Page-News list} \wedge \text{select}) \quad (2)$$

- The student role has the authority to seek for their courses’ scores, but no returning to the score management page to modify their scores.

$$AG(\text{Role-S} \rightarrow \neg(\text{return} \wedge EF\ \text{Page-course score})) \quad (3)$$

- The student cannot link the page of manager view page.

$$AG(\text{Role-S} \rightarrow \neg(\text{Link} \wedge EF\ \text{Page-M View})) \quad (4)$$

- The student cannot link the page of teacher view page.

$$AG(\text{Role-S} \rightarrow \neg(\text{Link} \wedge EF\ \text{Page-T View})) \quad (5)$$

Safety property of the teacher role includes:

- The teacher cannot link to the student information page to seek their private information.

$$AG(\text{Role-T} \rightarrow \neg(\text{link} \wedge EF\ \text{Page-S view}) \quad (6)$$

- The teacher has been authorized to browse the news page, but no adding the news page.

$$AG(\text{Role-T} \rightarrow \neg(\text{link} \wedge EF\ \text{Page-add News})) \quad (7)$$

- The teacher cannot select news in the News list page so that news can be deleted.

$$AG(\text{Role-T} \rightarrow \neg EF\ (\text{Page-News list} \wedge \text{select}) \quad (8)$$

- The teacher cannot link the page of the manager view page.

$$AG(\text{Role-T} \rightarrow \neg(\text{link} \wedge EF\ \text{Page-M View})) \quad (9)$$

Safety properties of the manager role include:

- The manager cannot link the page of the teacher view page.

$$AG(\text{Role-M} \rightarrow \neg(\text{link} \wedge EF\ \text{Page-T View})) \quad (10)$$

The manager cannot link the page of the student view page.

$$AG(\text{Role-M} \rightarrow \neg(\text{link} \wedge EF\ \text{Page-S View})) \quad (11)$$

4. Theories for Model Partition

According to safety properties of each role, we check the navigation model to obtain some counterexamples from the model checker. And then according to those counterexamples, we delete the ineffective navigation behaviors in navigation model and then it is partitioned a sub-model.

The frame of model partition based on model checking is shown in Figure 4. We input both the program of the navigation model described as the Kripke structure and a new safety property under verification in a model checker. If the navigation model satisfies this given property, the output of the model checker is true, otherwise the output is false and a counterexample trace is also outputted. When there is a counterexample trace in the model checker, the related ineffective navigation behaviors in navigation model/sub-model until the reduced model satisfies this safety property. If the output of the model checker is true, the next safety property will be inputted in the model checker. By repeating the above process, we can obtain a sub-model which meets to all safety properties of a user role. The detail partition steps are as follows.

(1) We input both a new safety property of the user role *u* and the navigation model of a Web application into the model checker;

(2) If a counterexample is outputted in the model checker, an ineffective navigation behavior will be deleted;

(3) Repeat (2) until the output is true;

(4) Delete all unreachable states from the initial state by manual and the related transitions in navigation model and then a navigation sub-model of the user role u is obtained.

We select a new user role and execute steps 1-4 again, then another sub-model of this user role can be obtained. We choose NuSMV as our model checker. So the Kripke structure needs to be translated into the NuSMV program. Since NuSMV only supports integer and enumeration types, we give some formal definitions of the user roles in the navigation model

so that the information of user roles for the states can be easily converted into the NuSMV program.

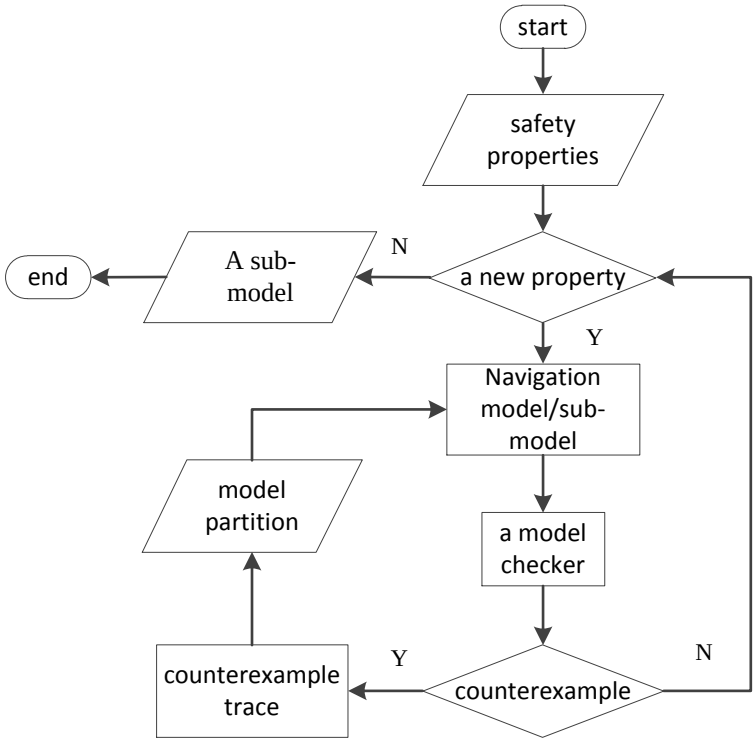


Figure 4. The Frame of Model Partition

Definition 4 (n-roles code). The n-roles code in pages is a binary number with the length n indicated n roles in WAs, where 1 denotes that a role has an authority to access pages and 0 represents that a role has not the authority to access pages. *e.g.*, for 3-roles code 011, there are three roles in pages, the first role has not the authority to visit pages and the second, and the third role has the authority to visit pages.

Definition 5 (basic code). The basic code is a sort of n-roles code which only denotes one role's code.

For the student course management system shown in Figure 2, there are three user roles. Therefore, there are the 3-roles code and three basic codes in pages of this system shown in Table 1.

Table 1. Three Basic Codes of the Student Course Management System in Figure 2

basic codes	001	010	100
user roles	Student role	Teacher role	Manager role

Definition 6 (hybrid code). A hybrid code, as a n-roles code, is the add operation result of the different base codes.

According to definition 6, we can set that the page is visited by two or more roles. For three basic codes, their hybrid codes are as follows.

011=001+010: denotes that roles 1 and 2 can visit the pages.

101=001+100: denotes that roles 1 and 3 can visit the pages.

110=100+010: denotes that roles 2 and 3 can visit the pages.

111=001+010+100: denotes that roles 1, 2 and 3 can visit the pages.

To obtain the effective navigation behaviors, we give the definition of role navigation function.

Definition 7 (role navigation function). Role navigation function is RN: $B \rightarrow C_b$, where B is a set of basic codes, b denotes a basic code and C_b is a union set of both $\{b\}$ and the set of the hybrid codes of b.

For a user, there is always a specific role contained in the pages. We use the decimal digits to denote the 3-roles codes of the user roles in Table 2.

Table 2. Decimal Digits of the 3-roles Codes

roles	role 1	role 2	role 3	roles 1 and 2	roles 1 and 3	roles 2 and 3	roles 1, 2 and 3
n-roles code	001	010	100	011	101	110	111
decimal digit	1	2	4	3	5	6	7

If we use integer to replace the 3-roles codes, role navigation function of this system is as follows:

- $RN(1)=\{1,3,5,7\}$ denotes role navigation function of role 1;
- $RN(2)=\{2,3,6\}$ denotes role navigation function of role 2;
- $RN(4)=\{5,6,7\}$ denotes role navigation function of role 3.

Theorem 1. For the basic code x and the hybrid code y of a Web application, $y \in RN(x)$ if and only if $x = x \wedge y$.

Proof (1) sufficient condition:

Since $y \in RN(x)$, $y = x$ or y is a hybrid code of x according to definition 7. For $y = x$, there exists $x \wedge y = x \wedge x = x$. For the latter, there must exist the basic codes $k_1, k_2, \ldots, k_i$ such that $y = x+k_1+k_2+\ldots+k_i$ according to definition 6. Suppose $k = k_1+k_2+\ldots+k_i$. Hence $y = x + k$. From definition 5, $x \wedge k = 00\ldots0$. Hence $y \wedge x = (x + k) \wedge x = x \wedge x + k \wedge x = x + 00\ldots0 = x$. Hence sufficient condition is proved.

(2) Necessary condition:

From definition 5, if the i-th bit of x is 1, other bits of x must be 0. Since $x = x \wedge y$, the i-th bit of y is also 1. And since y is a hybrid code, y is the result of add operation both x and other basic codes. From definition 5, add operation of among basic codes cannot change the location of 1 in the basic code. According to the definition 7, $y \in RN(x)$**.** Hence necessary condition is proved.

Theorem 1 is to provide a method to calculate role navigation function of all roles in a Web application.

Theorem 2. What a new role is added to the Web application cannot change decimal digits of all original n-roles codes in the Web application.

Proof:

(1) Suppose that there exist *i* roles in a WA. And then add a new role to this WA. From definition 4, the basic code of a new role is $100\ldots0_{i}$, and other basic codes are added one "0" in the far left of them. Hence decimal digits of all original basic codes do not change.

(2) And since the hybrid codes are the add operation result of other basic codes, decimal digits of the hybrid codes of all original basic codes do not also change.

In sum, decimal digits of all original n-roles codes are not changed when the system adds a new role.

Theorem 2 ensures that we can adopt the incremental method to verify a model, avoiding the state space explosion problem during model checking. Suppose the basic code *x* in the page A and the n-roles code y in the page B. If $y \in RN(x)$, the user with the basic code *x* can go to the page *B* from the page *A*.

Definition 8 (Role assignment operation). If the user can go to the page B with a hybrid code y from the page A with the basic code x, the n-roles code of the page B changes from y to x in the NuSMV program.

From definition 8, we can set role conversion operation of navigation model in the NuSMV program. *e.g.*, for navigation model in Figure 2, if the student role logins the system, the role of the state s_1 is rewritten as Role=S. That means Role=1 of s_1 in the NuSMV program. Therefore the role of the state s_2 changes from Role=T,S,M to Role=S during model checking according to definition 8..

In this paper, we ignore the detail descriptions of the NuSMV program of navigation model, and the related studies can be found in our previous work [8] and the research of Cimatti, *et al.*, [13].

Table 3. The Counterexample Traces of the Manager's Role

No.	The violated safety properties	Counterexample traces	Unsafe behaviors
1	AG(Role-M→¬(Link∧EF Page-T View))	< s_1, s_2, s_8 >	(s_2, s_8, link)
2	AG(Role-M→¬(Link∧EF Page-S View))	< s_1, s_2, s_{12} >	(s_2, s_{12}, link)

Table 4. The Counterexample Traces of the Teacher's Role

No.	The violated safety properties	Counterexample traces	Unsafe behaviors
1	AG(Role-T→¬(link∧EF Page-S view)	< s_1, s_2, s_{12} >	(s_2, s_{12}, link)
2	AG(Role-T→¬(Link∧EF Page-add News))	< s_1, s_2, s_8, s_4, s_5>	(s_4, s_5, link)
3	AG(Role-T→¬EF (Page-News list ∧select))	< s_1, s_2, s_8, s_4, s_7>	(s_4, s_7, select)
4	AG(Role-T→¬(Link∧EF Page-M View))	< s_1, s_2, s_3 >	(s_2, s_3, link)

Table 5. The Counterexample Traces of the Student Role

No.	The violated safety properties	Counterexample traces	Unsafe behaviors
1	AG(Role-S→¬(Link∧EF Page-M View))	< s_1, s_2, s_3 >	(s_2, s_3, link)
2	AG(Role-S→¬(Link∧EF Page-T View))	< s_1, s_2, s_8 >	(s_2, s_8, link)
3	AG(Role-S →¬(Link∧EF Page-add News))	< s_1, s_2, s_{12}, s_4, s_5>	(s_4, s_5, link)
4	AG(Role-S→¬EF (Page-News list ∧select)	< s_1, s_2, s_8, s_4, s_7>	(s_4, s_7, select)
5	AG(Role-S →¬(return∧EF Page-course score))	< s_1, s_2, s_{12}, s_{10}, s_9>	(s_{10}, s_9, return)

According to the counterexample traces and unsafe behaviors shown in Tables 3-5, we can obtain two sub-models shown in Figures 5-7, respectively.

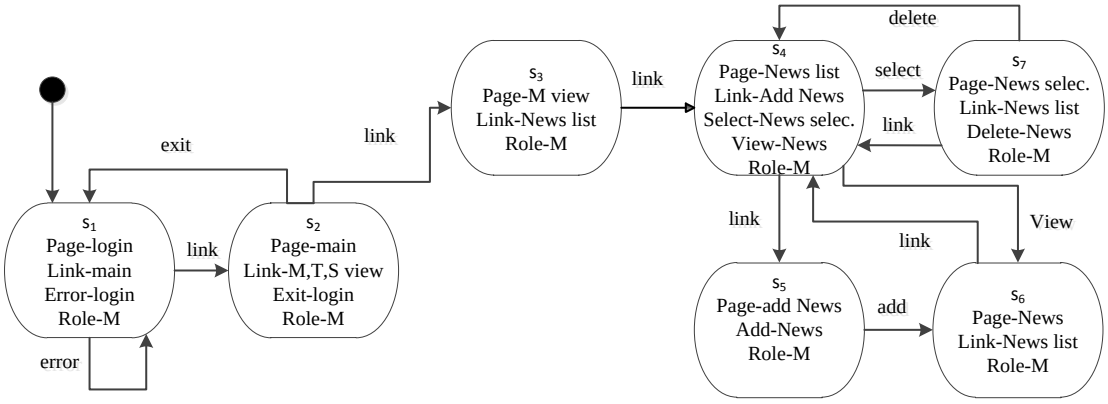


Figure 5. The Navigation Sub-model of the Manager's Role

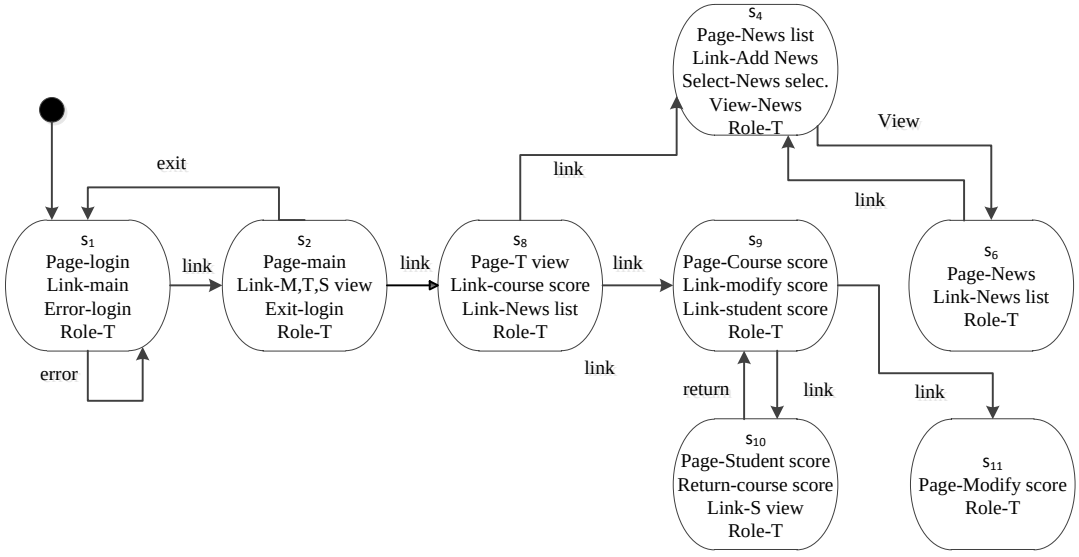


Figure 6. The Navigation Sub-model of the Teacher's Role

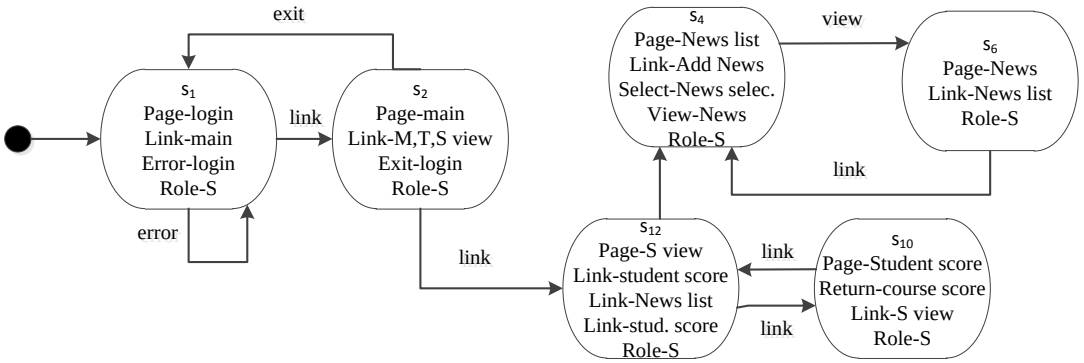
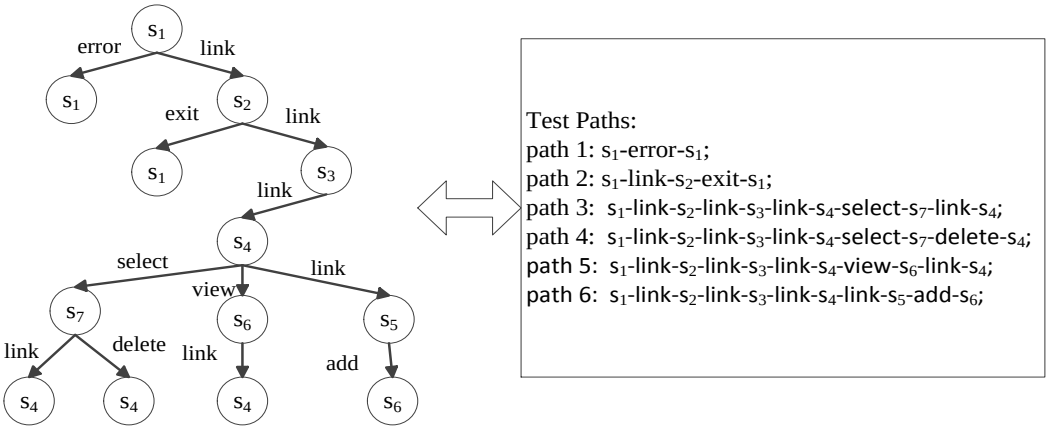


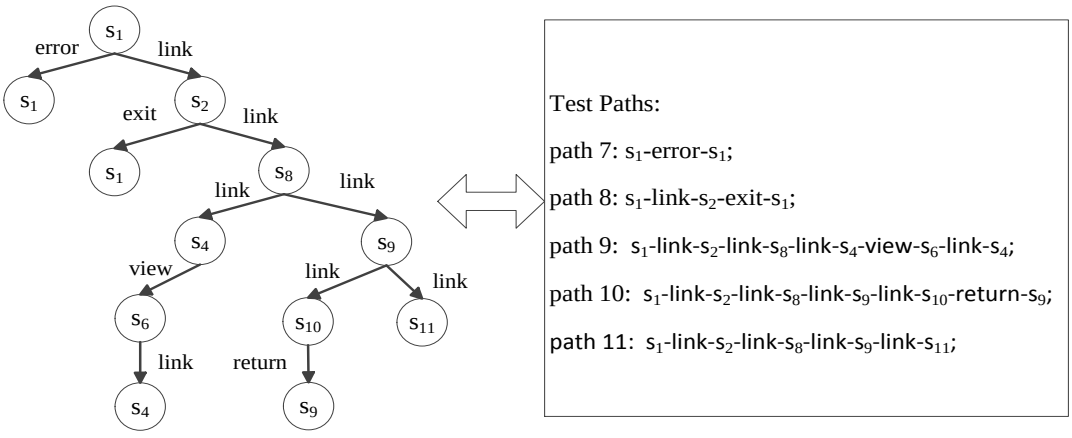
Figure 7. The Navigation Sub-model of the Student Role

5. Test Generations and Redundancy Reduction

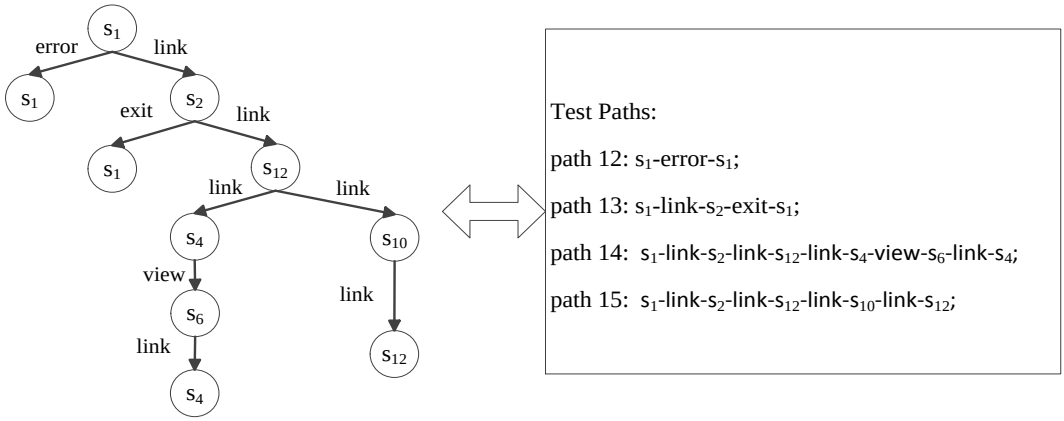
In this section, we employ the branch-first search method to traversal three sub-models. And then three test trees are generated from them, respectively, where the test tree and its test paths of the manager's role is shown in Figure 8 (a), the teacher's role is shown in Figure 8 (b) and that of the student role is shown in Figure 8 (c). It is easy to confirm that all test paths are effective.



(a)



(b)



(c)

Figure 8. Test Tree and Test Paths of Three Navigation Sub-models

There are many redundant sequences among test paths, such as paths 1, 7 and 12. For this reason, the related redundancy reduction techniques need to be implemented into our approach.

Software testing is to satisfy specify test intention or test coverage criterion. If test intention is state coverage [2, 14] for a test, we need to select test paths, which cover the most uncovered states. So test paths 3, 6, 10, 11 and 15 are selected. If test intention is transition coverage [2, 14], we need to delete all 1 to 1 redundant test paths from all test paths. Therefor test paths 1, 2, 3, 4, 5, 6, 9, 10, 11, 14 and 15 are selected. Based on this reduction way, the reduced test paths will hold the same fault detection capability with the original test paths for the same purpose.

6. Related Work

Testing web applications have been highlighted for the past decade. And a number of modeling and test generation techniques for Web applications have also been already proposed, each of which has different origins and pursuing different test goals for dealing with the unique characteristics of WAs. Here we briefly describe some relevant studies.

Authors in [15] developed a tool ReWeb to perform analyses on web sites and introduced a graphical representation of the web site to extend to traditional static flow analyses for web applications. The method proposed in [16] was based on a UML model of Web applications and considered the testing and validation of the developed web system. However, both methods lack of the discussions of effective redundancy reduction.

Subraya and Subrahmanya [17] proposed object driven performance testing. They illustrated a new testing process that employs the concept of decomposing the behavior of a Web application into testable components. Different from theirs, our approach constructs the navigation model of a Web application by the Kripke structure. Therefore, the navigation model can be verified safety properties of a WA by a model checker.

Benedikt, *et al.*, [18] built a dynamic navigation test tool (named as VeriWeb) for Web applications. VeriWeb explores sequences of links in Web applications by nondeterministically searching action sequences, starting from a given URL. The test method inVeriWeb is based on graphs where nodes are Web pages and edges are explicit HTML links, and the size of the graphs is controlled by a pruning process. However, this test method cannot consider the effectiveness of test paths.

A system-level testing technique that combines test generation based on FSM with constraints is proposed by A. Andrews, *et al.*, [1]. And they were modeling Web applications by hierarchies of an FSM which used to model subsystems of Web applications. Test sequences are generated based on FSMs and they try to use input constraints to restrict the state space explosion. However, the hierarchies' idea in [1] is realized by manual way and is hardly realized in real implementation when a Web application becomes very complex.

Miao, *et al.*, [19] took special care on Web browser interactions during the user's traversal within the hypermedia space in order to specify possible inconsistencies between Web browser interfaces and user cognitions. Jessica, *et al.*, [20] provided a definition of Web model in terms of labeled transition systems by incorporating the abstract behavioral model of the Web browsers. However, the test generation based on the presented model is not given out.

Song, *et al.*, [21] and Chen, *et al.*, [22] paid special care on Web browser's interactions and proposed an approach to modeling on-the-fly navigation models and test generation with Web browser interactions and PDR. The process of redundancy reduction of test sequences is to make use of the overlap among test sequences to link them. However, this approach does not consider the effectiveness of redundancy reduction.

Our previous work [8] focuses on testing and can be combined with the usual Kripke structure into windows, links, pages and actions to achieve automated verification and test of Web applications. In this paper, we lay emphasis on the process of test optimization generation by model checking.

In [23], we proposed the regular expression-based test generation method. This method employs the regular expression to set up the model of Web applications and then decomposes regular expression of a model into a set of sub-expressions. Finally, test cases are derived from those sub-expressions. Different from our earlier work, this paper focuses on test generation based on model partition. In our previous work [24], we discuss the method of model partition-based testing for Web applications, while we supplement and complete the related theories of model partition and the discussion of redundancy reduction of test paths in this paper.

7. Conclusion

We propose a formal approach to test generation of Web applications by using model partition. Our approach consists of three phases. The first phase is to set up navigation model based on the Kripke structure. In the second phase, we put forward the method of model partition via the model checking technique. In the third phase, we construct a test tree for each sub-model and then generate test paths. Finally, we reduce test paths based on the special test intention.

In the paper, we do not care about the process of the real test case generation, which can be realized by the method of test refinement [25]. In the future, we plan to set up the navigation model of the interactions of between browser and users, and then generate a set of optimized test paths based on our approach in this paper.

Acknowledgements

This work is supported by National Natural Science Foundation of China (NSFC) under grant No. 61170044 and 60970007, Project of Science and Technology Commission of Shanghai Municipality under Grant No. 10510704900 and Shanghai Leading Academic Discipline Project (Project Number: J50103).

References

[1] A. Andrews, J. Offutt and R., Alexander, "Testing web applications by modeling with FSMs", Softw. Syst. Model, vol. 4, no. 3, **(2005)**.
[2] P. Liu, H. -k. Miao, H. -W. Zeng and Y. Liu, "FSM-based testing: Theory, method and evaluation", Chinese Journal of Computers, vol. 34, no. 6, **(2011)**.
[3] F. Ricca and P. Tonella, "Analysis and testing of Web applications", Proceedings of the 23rd International Conference on Software Engineering, **(2001)** May 12–19; Toronto, Ontario, Canada.
[4] C. Bellettini, A. Marchetto and A. Trentini, "Testuml: User-metric driver web applications testing", Proceedings of 12th ACM Symposium on Applied Computing **(2005)** March 13-17; New Mexico, USA.
[5] S. Elbaum, G. Rothermel, S. Karre and M. Fisher, "Leveraging user session data to support web application testing", IEEE Trans. Softw. Eng., vol. 31, no. 3, **(2005)**.
[6] A. Marchetto, P. Tonella and F. Ricca, "State-based testing of ajax web applications", Proceedings of the 3rd International Conference on Software Testing Verification and Validation, **(2008)** April 9-11; Lillehammer, Norway.
[7] M. Huth and M. Ryan, "Logic in Computer Science, Modelling and Reasoning about Systems", Cambridge University Press, England, Cambridge, **(2004)**.
[8] H. W. Zeng and H. K. Miao, "Model Checking-Based Testing of Web Applications", Wuhan University Journal of Natural Sciences, vol. 12, no. 5, **(2007)**.
[9] K. L. McMillan, "The SMV System for SMV version 2.5.4", **(2006)** October, http://www.cs.cmu.edu/~modelcheck/smv/-smvmanual.ps.
[10] G. J. Holzmann, "The Model Checker SPIN", IEEE Transactions on Software Engineering, vol. 23, no. 5, **(1997)**.
[11] A. Gargantini and C. L. Heitmeyer, "Using Model Checking to Generate Tests from Requirements Specifications", In Proceedings of Joint 7th European Software Engineering Conference and 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering, **(1999)** September 6-10; Toulouse, France.
[12] K. Gopinath, Jon Elerath, D. Long, K. Gopinath, J. Elerath and D. Long, "Probabilistic model checking and reliability of results", Proceedings of 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, **(2008)** April 16-18; Bratislava, Slovakia.
[13] A. Cimatti, E. Clarke, F. Giunchiglia and M. Roveri, "NuSMV: A New Symbolic Model Verifier", COMPUTER AIDED VERIFICATION, Lecture Notes in Computer Science, Springer, **(1999)**.
[14] P. Liu, H. -K. Miao, H. -W. Zeng and J. Mei, "DFSM-based minimum test cost transition coverage criterion", Journal of Software, vol. 22, no. 7, **(2011)**.
[15] F. Ricca and P. Tonella, "Analysis and testing of Web applications", In Proceedings of the 23rd International Conference on Software Engineering, **(2001)** May 13-18; Toronto, CA.
[16] P. Tonella and F. Ricca, "Testing processes of web applications", Annals of software engineering, vol. 14, no. 1, **(2002)**.
[17] B. M. Subraya and S. V. Subrahmanya, "Object Driven Performance Testing of Web Applications", The First Asia-Pacific Conference on Quality Software, **(2000)** October 30-31; Hong Kong, China.
[18] M. Benedikt, J. Freire and P. Godefroid, "VeriWeb: Automatically Testing Dynamic Web Sites", In Proceedings of 11th International World Wide Web Conference, **(2002)** May 7-11; Honolulu, Hawaii, USA.
[19] H. K. Miao, Z. S. Qian and T. He, "Modeling Web Browser Interactions Using FSM", Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, **(2007)** December 11-14; Washington, DC, USA.
[20] J. Chen and X. Zhao, "Formal Models for Web Navigations with Session Control and Browser Cache", In Proceedings of the 6th International Conference on Formal Engineering Methods, **(2004)** November 8-12; Seattle, WA, USA.
[21] B. Song and H. Miao, "Modeling Web Applications and Generating Tests: A Combination and Interactions guided Approach", In Proceeding of the Third IEEE International Symposium on theoretical Aspects of Software Engineering, **(2009)** July 4-6; Beijing, China.
[22] S. Chen, H. Miao, B. Song and Y. Chen, "Towards Practical Modeling of Web Applications and Generating Tests", In Proceedings of the Fourth International Symposium on Theoretical Aspects of Software Engineering, **(2010)** August 25-27; Taipei, Taiwan.
[23] P. Liu and H. Miao, "A new Approach to Generating High Quality Test Cases", In Proceedings of IEEE 19th Asian Test Symposium, **(2010)** December 1-4; Shanghai, China.
[24] P. Liu, H. Miao, H. Zeng and L. Cai, "Model Parition-based testing for Web Applications", In Proceedings of the second international conference on computers, networks, systems, and industrial applications, **(2012)** July 16-18; Phoenix Island, Jeju Island, Korea.
[25] J. Derrick and E. A. Boiten, "Testing Refinements of State-based Formal Specifications", Software Testing, Verification & Reliability, vol. 9, no. 1, **(1999)**.

Authors

Pan Liu received the MSc degree in computer software and theory from Nanchang University in 2006 and the PhD degree in computer application from Shanghai University in 2011. Now he is a lecturer at college of computer engineering and science, Shanghai Business School, Shanghai, China. He is also an associate research in Shanghai Key Laboratory of Computer Software Testing & Evaluating, Shanghai, China. His papers have been published in some well-known international Journals and some IEEE conferences. His main interests include software testing, model-based testing, formal method, and algorithm design.

Huaikou Miao is a professor and a doctoral supervisor at school of computer engineering and science at Shanghai University and is a senior member of CCF in China. He was a program chair of ICIS 2009 and has been invited as a keynote at several well-known international conferences. More than one hundred his papers have been published in international journals and international conferences. His interests are software engineering, software testing, model checking, and formal method.

Hongwei Zeng is a researcher and a doctoral supervisor at school of computer engineering and science at Shanghai University and is a senior member of CCF in China. He received the PhD degree in computer application from Shanghai University in 2008. He was a program chair of CNSI 2012. More than twenty his papers have been published in some journals and international conferences. His interests are software engineering, software testing, and model checking.

Lizhi Cai is a researcher and a master supervisor at East China University of Science and Technology, and is a director of Shanghai Key Laboratory of Computer Software Testing & Evaluating in China. He received the PhD degree in computer application from Shanghai University in 2009. He has been published more than ten his papers in some international conferences. His interests are software testing and test standard design.