基于Ajax技术的Web应用的建模与测试用例生成

贺 涛[1,2] 缪淮扣[1,2] 钱忠胜[3]
(上海大学计算机工程与科学学院 上海 200072)[1] (上海市计算机软件评测重点实验室 上海 201114)[2]
(江西财经大学信息管理学院 南昌 330013)[3]

摘 要 Ajax技术使Web应用能够通过异步请求从服务端获取数据,并在网页上局部刷新显示。这使得一张网页可以包含多个不同状态,状态数的激增使其关系变得更加复杂,给Web应用的建模与测试带来了更大的难度。研究基于Ajax技术的Web应用的建模与测试用例生成方法,给出一种可行的产生测试用例的技术。结合课题组自身开发的项目进行建模与测试用例的生成分析,结果表明,该技术能有效地得到所需的测试用例。

关键词 Web应用,模型检查,测试用例,Ajax技术,Kripke结构
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.08.047

Modeling and Test Case Generation for Ajax-based WA

HE Tao[1,2] MIAO Huai-kou[1,2] QIAN Zhong-sheng[3]
(School of Computer Engineering & Science, Shanghai University, Shanghai 200072, China)[1]
(Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201114, China)[2]
(School of Information Technology, Jiangxi University of Finance & Economics, Nanchang 330013, China)[3]

Abstract Ajax technology enables Web applications to get data from the server through an asynchronous request, and partially refresh the Web page. This allows a Web page contains multiple different states. The sharp increase of the number of states makes the Web applications more complicated and brings greater difficulty to modeling and testing of Web applications. We researched modeling and test case generation method of Web applications and gave a feasible technology to generate test case. Finally, we verified the method combining with the project developed by our research group. And according to results of verification, it can generate test case effectively.

Keywords Web application, Model checking, Test case, Ajax, Kripke

1 引言

Ajax是Asynchronous JavaScript and XML(以及DHTML等)的缩写。它作为Web2.0领域非常热门的客户端技术,于2005年在Adaptive Path公司的Jesse James Garret的一篇名为《Ajax:一种Web应用程序开发的新方法》的论文中提出[1]。使用基于Ajax技术开发的Web应用拥有更好的用户体验和更强的用户交互能力,能节约更多的网络资源。正因为Ajax有这么多的优点,它才得到了快速的发展,并且确立了在Web领域不可动摇的地位,推动了互联网时代的到来。

传统的Web应用中,用户首先点击网页中某个元素或者在浏览器地址栏输入URL向Web服务器发送请求,然后等待服务器对用户请求的解析,最后服务器返回相应的结果。在这个请求的过程中,客户端必须等待服务器的响应结果。这种"请求-等待-请求"模式带给了用户极差的交互体验。Ajax采用的是异步交互的方式,当用户向服务器发出请求后,客户端不必等待服务器的响应,就可以进行其它的操作或者发出其它的请求,服务器会处理用户的请求返回结果,最后无刷新地显示在网页中,这大大改善了用户的交互体验。目前,Ajax技术已经在Google suggest、Gmail、Amazon、Yahoo等很多或大或小的Web应用中使用。

但是,基于Ajax技术的Web应用比传统的Web应用复杂了很多,这给Web应用的测试带来了巨大的挑战。本文使用模型检验的方法研究基于Ajax技术的Web应用的建模和测试用例生成。在对传统的Web应用进行模型检验时,建模的方法一般是将Web模型中的每一张网页作为模型中的一个状态,状态之间通过超链接、表单提交、构件调用等同步的请求方式进行迁移[2]。而对于基于Ajax技术的Web应用,一个网页中会有多个HTML元素触发Ajax请求,导致网页中的DOM(Document Object Model)结构发生局部的变化,因此每张网页会包含多个状态。状态之间的迁移可能是通过超链接、表单提交、构件调用等同步请求引起的网页之间的跳转,也可能是Ajax异步请求引起的网页内部的变化。基于上

到稿日期:2013-10-06 返修日期:2014-01-21 本文受国家自然科学基金项目(61170044,61073050,61262010),上海市自然科学基金(13ZR1429600),上海高校青年教师培养资助计划(ZZSD13008)资助。

贺 涛(1989—),男,硕士生,主要研究方向为软件测试;缪淮扣(1953—),男,教授,博士生导师,主要研究方向为软件工程、软件形式化、软件测试、模型检验;钱忠胜(1977—),男,副教授,主要研究方向为软件自动化、软件测试、社会化网络、电子商务、物流信息化。

述特点，本文提出将 Web 应用的一张网页划分为能局部刷新的各个部分，根据这些部分的内容的变化将网页分成多个不同的状态，网页内部的状态迁移由 Ajax 请求触发，使用 UML 状态图对应用建模，将得到的模型用 Kripke 结构表示。用 CTL 公式来描述模型的陷阱性质，通过将陷阱性质和模型程序输入到模型检验工具 SMV 中执行，得到反例并转化为测试用例。

2 相关工作

基于模型的测试思想来源于 1956 年 Moore 对自动机的阐述。到了 20 世纪 80 年代，基于模型的测试方法、测试覆盖准则被相继提出并率先在协议测试方面得到应用。90 年代，Callaha 和 Engels 首先使用模型检验器来自动产生测试用例。近年来，结合不同模型及技术的混合测试方法正在兴起。一些基于模型的测试支持工具或工具原型也被开发出来。到目前为止，基于模型的测试是应用模型检验技术的自动化测试的一个重要研究方向。模型检验(model checking)[3]是一种形式化验证技术，通过状态空间搜索来检验给定的模型是否符合某些验证性质，不满足时会输出反例(counter example)，根据反例来构造测试用例并实例化。

软件测试使用的建模方法有：有限状态机、UML 模型和马尔可夫链等[4,10]。有限状态机可以用状态迁移图或状态迁移矩阵表示，可以根据状态覆盖或迁移覆盖产生测试用例。基于 UML 模型的测试研究主要集中于 UML 的状态图，状态图是有限状态机的扩展，强调了对复杂实时系统进行建模，提供了层次状态机的扩展。马尔可夫链是一种以统计理论为基础的统计模型，可以描述软件的使用，在软件统计测试中得到了广泛应用。

使用模型检验进行测试的方式主要有两种。一种是根据现有的覆盖准则，利用 CTL、LTL 对待测模型设计陷阱性质，通过检验陷阱性质输出反例。文献[5]利用节点覆盖、边覆盖、边组合覆盖来产生陷阱性质。文献[6]利用分支覆盖准则产生陷阱性质。另一种是利用变异分析产生反例，可以对模型或者性质进行变异的处理，变异测试需要选择合适的变异算子。文献[7]提出把 VRO(Variable Replacement Operator)变异算子作用在 SMV 建模输入上。

近年来，随着互联网行业的发展，模型检验生成测试用例的方法在 Web 领域也得到了应用，文献[2,5]利用模型检验对 Web 应用建模测试。针对传统的 Web 应用的网页之间发生的请求、跳转，利用 UML 状态图对 Web 应用建模并用 Kripke 结构表示，根据现有的覆盖准则利用 CTL 计算树逻辑进行陷阱性质的设计，通过 SMV 产生反例。

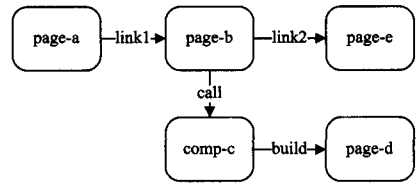
随着 Web2.0 时代的到来，Ajax 技术得以流行，几乎所有的 Web 应用都用到 Ajax。文献[9]提出了基于状态的 Ajax 的 Web 应用的测试方法，Ajax 操纵的网页 DOM 被抽象到一个状态模型，Ajax 请求的回调函数执行被关联状态的迁移，由状态模型的语义交互事件产生测试用例。文献[10]中使用了状态流图对 Ajax 的应用进行建模。本文根据网页的 DOM 结构将其划分为不同的会发生局部刷新的子部分，子部分的变化导致整张网页内的状态的迁移。

3 基于 Ajax 技术的 Web 应用建模

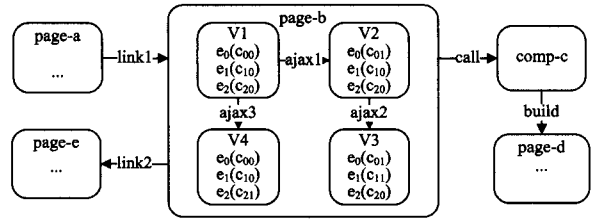
UML 状态图(State Chart Diagram)是在有限状态机的基础上引入状态的层次化、并发以及互斥来提高自身的描述能力，所以状态图是有限状态机的一种扩展[6]。它可以通过对象的状态以及状态间的转换来描述一个对象基于事件反应的动态行为。本文采用 UML 状态图对 Web 应用建模。

在 Ajax 技术出现之前的 Web 系统的建模过程中，通常的方法是将一张网页、一个构件等表示为 Web 模型的一个状态[5]。当发生超链接、表单提交、构件调用等同步请求时，会触发当前网页跳转到下一个网页或者由一个构件跳转到一个网页或者另一个构件，从而引起 Web 模型的状态迁移。

如图 1(a)所示，网页 page-a 通过超链接 link1 跳转到网页 page-b，而 page-b 可以通过超链接 link2 跳转到网页 page-e，或者通过调用构件 comp-c，根据构件 comp-c 的执行结果进入网页 page-d。



(a) 传统Web应用建模



(b) 基于Ajax技术的Web应用建模

图 1 Web 应用的建模

而在基于 Ajax 技术的 Web 应用中网页内通常会包含异步的请求。通常，Ajax 请求的结果只是引起原网页内局部内容的刷新并不会使原网页跳转到其它网页或构件。当网页内有多个不同 Ajax 请求分别引起网页内不同部分内容的局部刷新时，其中的一个请求发生后就会造成整张网页的内容与原网页内容不同，网页内显示的内容会因为不同的 Ajax 请求的发生产生不同的组合结果。

如图 1(b)所示，网页 page-a 通过超链接的方式跳转到网页 page-b，在网页 page-b 中有 3 个部分(e_0，e_1，e_2)是包含 Ajax 操作的，而这些部分的初始内容分别是(c_{00}，c_{10}，c_{20})，当发生 ajax1 请求后网页 page-b 中的 e_0 部分的内容由 c_{00} 变为 c_{01}，网页 page-b 的状态由 V_1 变为 V_2，同样地，当 ajax2 和 ajax3 操作发生时网页 page-b 中会有相应的部分的内容发生改变，引起网页内的状态迁移。因此，不能使用传统的 Web 系统的建模方法简单地将 Web 系统导航中的每一张网页、每一个构件表示为一个状态，状态之间只通过同步请求的方式进行迁移。

为了根据基于 Ajax 技术的 Web 应用的特点实现对它的建模，以更好地描述异步请求前后网页内部状态的变化，本文提出将 Web 应用的一张网页划分为能局部刷新的各个部分，根据这些部分的内容的变化将网页分成多个不同的状态进行建模，并给出以下规则。

规则 1(网页状态元素) 将会随着 Ajax 请求产生的结果局部刷新内容的部分定义为网页状态元素 e_i，网页状态元素可以是网页中局部的信息、表格数据、超链接、排序方式等

所有可以被动态修改的内容，一张网页中包含的所有网页状态元素的集合为 $e=\{e_0,e_1,\cdots,e_i,e_{i+1},\cdots\}$。

如图 1(b)所示，e_0,e_1,e_2 就代表网页状态元素。这些部分内容的变化会直接导致整张网页的变化。

规则 2(网页状态元素 Ajax 操作集) 将能引起网页状态元素 e_i 内容发生变化的 Ajax 请求操作定义为 $ajax\text{-}op_{ij}$，其中下标 i 表示第 i 个网页状态元素，j 表示在 e_i 上的第 j 种 Ajax 请求操作。某个网页状态元素 e_i 的所有 Ajax 请求操作集合可以表示为 $ajax\text{-}op_i=\{ajax\text{-}op_{i0},ajax\text{-}op_{i1},\cdots,ajax\text{-}op_{ij},ajax\text{-}op_{ij+1},\cdots\}$。

规则 3(网页状态元素值域) 由于网页状态元素的内容会随着在它之上的 Ajax 请求而改变，因此每个网页状态元素会有一个值域。我们将网页状态元素 e_i 的值域描述成 $c_i=\{c_{i0},c_{i1},\cdots,c_{ij},c_{ij+1},\cdots\}$，$c_{ij}$ 下标的 i 表示网页的第 i 个网页状态元素 e_i，j 表示 e_i 的第 j 种取值。另外，规定网页状态元素的特殊取值 Null，当取值为 Null 时表示网页在当前状态下不显示该网页状态元素。

规则 4(网页内子状态) 网页状态元素 e_i 上发生 Ajax 请求后，e_i 的内容就会发生改变，从而网页的状态发生变化。将网页 page-v 根据网页状态元素取值的不同组合划分为不同的子状态。一张网页的子状态集就表示为 $v=\{v_0,v_1,\cdots,v_i,v_{i+1},\cdots\}$。

规则 5(网页内子状态迁移) 将网页 page-v 所处的状态表示为它包含的所有网页状态元素的不同取值组合，如图 2 所示。网页 page-v 内子状态的迁移就可以表示成：

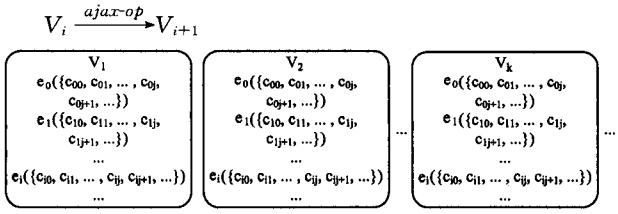


图 2 网页 page-v 包含的所有状态

4 实例分析

4.1 系统简介

本文采用本项目组开发的 Web 应用 CPMISS (Condominium Property Management Information Server Secretariat)作为实例研究基于 Ajax 技术的 Web 应用的测试。CPMISS 系统是一个小区物业管理公共信息平台，它包含注册校验、业户中心、用户信函、社区楼宇、会议中心等功能模块，此系统旨在降低物业管理成本，提高管理效率，营造智慧社区。本文主要针对 CPMISS 系统其中的一个功能模块——用户信函模块进行研究。用户信函模块主要为用户提供日常信函的收发、查看等功能，整个模块主要由 Ajax 技术实现。用户在 CPMISS 系统首页 page-Home 登入后，点击“用户信函”的超链接进入用户信函的主页 page-MailCenter。用户信函主页包含了信函模块的主要功能，包括收件夹、写信等，用户通过点击网页中的按钮发送 Ajax 请求局部刷新页面的方式进入不同的功能界面。其中收件夹可以以升序或降序的方式分页查看用户收到的信函，写信操作可以进入写信状态并发送信函。另外，用户信函主页以及跳转到用户中心页面的超链接，点击之后即离开用户信函页面进入用户中心页面 page-UserCenter。

4.2 系统建模

根据规则 1 可以确定用户信函模块 page-MailCenter 网页中的所有网页状态元素(focus, sort, pageIndex, sendResult)：

• focus 表示 page-MailCenter 网页当前聚焦的是哪个功能模块。

• sort 表示 page-MailCenter 网页展示信函列表的排序方式。

• pageIndex 表示 page-MailCenter 网页展示的信函列表当前处于哪一页。

• sendResult 表示发送信函的结果。

根据规则 3 可以确定用户信函模块网页 page-MailCenter 中的各个网页状态元素的值域：

• focus 的值域是(Home, Inbox, Write)，focus-Home 表示用户信函当前聚焦的是用户信函页面的主页，focus-Inbox 表示用户信函页面当前聚焦的是收件夹界面，focus-Write 表示用户信函主页当前聚焦的是写信界面。

• sort 的值域是(Asc, Desc, Null)，sort-Asc 表示信函以时间升序排列，sort-Desc 表示信函以时间降序排列，sort-Null 表示用户信函当前状态下没有对信函的排序操作。

• pageIndex 的值域为(First, Middle, Last, Null)。pageIndex-First 表示用户信函页面展示的是信函列表的首页，pageIndex-Middle 表示用户信函页面展示的是信函列表的首页和尾页之间的中间页，pageIndex-Last 表示用户信函列表展示的是信函列表的尾页，pageIndex-Null 表示用户信函页面当前状态下没有分页的操作。

• sendResult 的值域是(Success, Fail, Null)。其中，sendResult-Success 表示的是信函已发送成功，sendResult-Fail 表示的是信函发送结果失败，sendResult-Null 表示用户信函页面当前状态下不显示信函发送结果。

根据规则 4 和规则 5 可以确定用户信函模块 page-MailCenter 网页所有可能到达的状态，如图 3 所示。

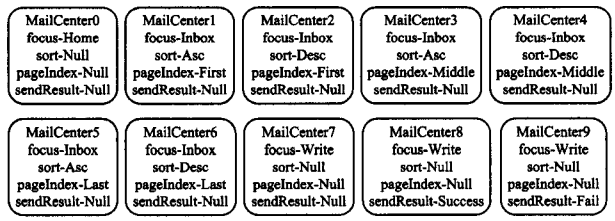


图 3 用户信函模块主页的状态

根据规则 2 可以确定引起网页状态元素值发生变化的 Ajax 请求操作：

• 引起 focus 变化的操作：(ajax-Inbox, ajax-Home, ajax-Write)。

• 引起 sort 变化的操作：(ajax-Desc, ajax-Asc)。

• 引起 pageIndex 变化的操作：(ajax-Prev, ajax-Next)。

• 引起 sendResult 变化的操作：(ajax-Send)。

根据规则 5 和 UML 状态图我们对 CPMISS 系统中用户信函模块建模，如图 4 所示。图中 page-Home, page-Login, page-UserCenter 是不包含 Ajax 请求的网页。call-LoginCheck 是登入校验构件的调用。page-MailCenter 网页包含的子状态 MailCenter0 到 MailCenter9 代表用户信函主页的可能状态，它们之间的迁移代表因 Ajax 请求导致的页面内部状态的变化。然后根据系统的状态图的描述，我们将其转化为 Kripke 结构模型，如图 5 所示。

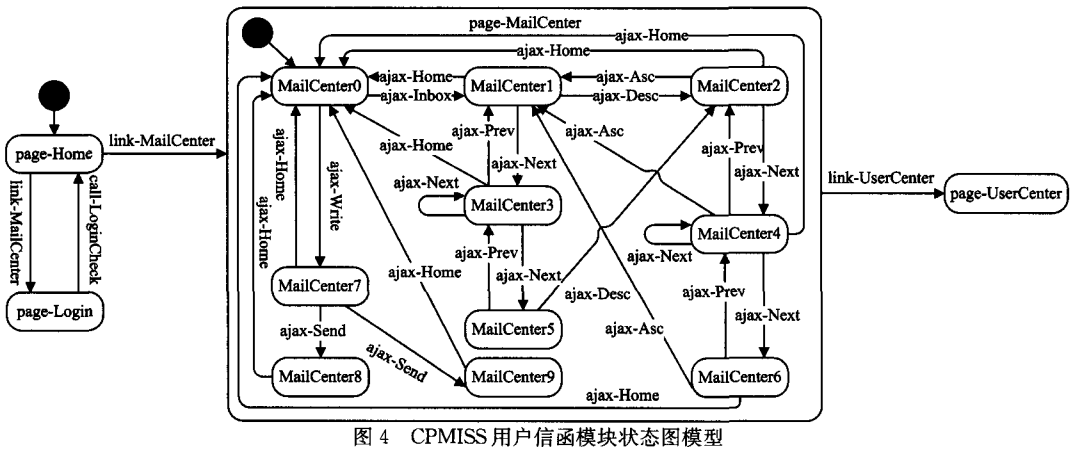


图 4　CPMISS 用户信函模块状态图模型

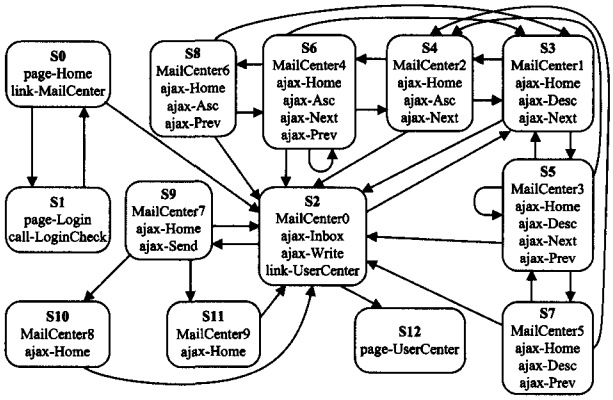


图 5　CPMISS 用户信函模块的 Kripke 结构模型图

4.3　陷阱性质描述

在对 CPMISS 系统的用户信函模块的模型进行验证时，我们使用计算树逻辑(CTL)来设计系统的陷阱性质[13]，分别对系统的可达性、活性、安全性进行检验。

系统可达性指系统中的每一个状态节点都至少能被访问到一次。本文建立的模型中，要求包括网页内子状态在内的所有状态都能被访问到。我们用状态覆盖准则来检验系统的可达性。对于没有 Ajax 请求的网页 page-v，一个网页可以表示为一个状态，用 CTL 公式表示其可达性性质：EFpage-v。对于包含 Ajax 请求的网页，将它表示为多个状态，其中的某一状态的可达性性质可以描述为：$EF\ V_i(i=1,2,\cdots,n)$。为了在模型检验器中得到可用于验证系统可达性的反例，我们将相应的陷阱性质描述成：$AG\neg page\text{-}v$，$AG\neg V_i(i=1,2,\cdots,n)$。

系统的活性指系统中的每一条迁移都至少能被访问到一次，即系统实现了所有合法的导航。我们用迁移覆盖准则来检验系统的活性。对于通过超链接方式导致网页之间状态的迁移($page\text{-}u$，$link$，$page\text{-}v$)，用 CTL 公式表示其活性：$AG(page\text{-}u\rightarrow(link\text{-}v\wedge EX\ page\text{-}v))$。对于通过 Ajax 请求方式导致网页内部状态的迁移(V_i，$ajax$，V_j)，用 CTL 公式表示其活性：$AG(V_i\rightarrow(ajax\wedge EX\ V_j))$。相应的陷阱性质描述成：$AG(page\text{-}u\rightarrow\neg(link\text{-}v\wedge EX\ page\text{-}v))$，$AG(V_i\rightarrow\neg(ajax\wedge EX\ V_j))$。

系统的安全性指系统不存在非法的导航。我们用迁移组合覆盖准则来验证系统的安全性。对于以超链接方式的网页之间的迁移，page-u 所能到达的其它网页用 $page\text{-}v_i(i=1,2,\cdots,k)$表示，page-u 的所有合法迁移可以用 CTL 表示为：$AG(page\text{-}u\rightarrow AX(page\text{-}v_1\vee page\text{-}v_2\vee\cdots\vee page\text{-}v_n))$，相应的验证性质表示为：$AG(page\text{-}u\rightarrow\neg EX\neg(page\text{-}v_1\vee page\text{-}v_2\vee\cdots\vee page\text{-}v_n))$。对于以 Ajax 请求方式的网页内部状态之间的迁移，状态 V_i 能迁移到其它状态用 $V_j(j=1,2\cdots,k)$表示，V_i 的所有合法迁移可以用 CTL 表示为：$AG(U_i\rightarrow AX(V_1\vee V_2\vee\cdots\vee V_n))$，如图 6 所示，相应的验证性质表示为：$AG(U_i\rightarrow\rightarrow EX\rightarrow(V_1\vee V_2\vee\cdots\vee V_n))$。

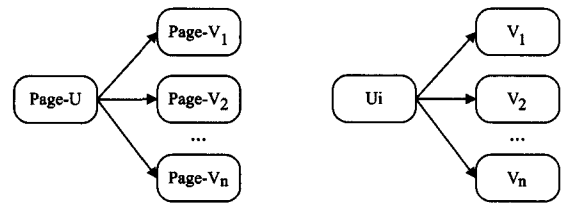


图 6　网页的合法迁移

4.4　测试用例生成

模型检验器 SMV 是美国 Carnegie Mellon 大学于 1992 年开发出的模型检验工具软件。SMV 系统采用符号模型检验(symbolic model checking)技术，以二叉图表示状态迁移关系，以计算不动点的方法检测状态的可达性和其所满足的性质，同时支持 CTL 或 LTL 性质的验证[12]。用 SMV 程序描述 CPMISS 系统的 Kripke 结构模型，如图 7 所示，结合 CTL 公式描述的可达性、活性、安全性的陷阱性质，使用 SMV 来生成反例。

```
MODULE main
VAR
state:{S0,S1 S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12};
page-Home:boolean;
link-MailCenter:boolean;
call-LoginCheck:boolean;
MailCener0:boolean;
MailCenter1:boolean;
MailCenter2:boolean;
MailCenter3:boolean;
ajax-Home:boolean;
ajax-Inbox:boolean;
.....
ASSIGN
    init(state):=s0;
    init(page-Home):=TRUE;
    init(link-MailCenter):=TRUE;
    init(call-LoginCheck):=FALSE;
    init(MailCenter0):=FALSE;
    init(MailCenter1):=FALSE;
```

```
    init(MailCenter2) := FALSE;
    init(ajax-Home) := FALSE;
    init(ajax-Inbox) := FALSE;
    ...
next(state) := case
    state=S0:{S1,S2};
    state=S2:{S3,S9,S12};
    state=S3:{S2,S4,S5};
    state=S4:{S2,S3,S6};
    ......
    TRUE:state;
esac;
next(MailCenter0) := case
    next(state)=S2:TRUE;
    TRUE:FALSE;
esac;
next(MailCenter1) := case
    next(state)=S3:TRUE;
    TRUE:FALSE;
esac;
next(ajax-Inbox) := case
    next(state)=S2:TRUE;
    TRUE:FALSE;
esac;
next(ajax-Asc) := case
    next(state)=S4|next(state)=S6|next(state)=S8:TRUE;
    TRUE:FALSE;
esac;
next(ajax-Send) := case
    next(state)=S9:TRUE;
    TRUE:FALSE;
esac;
next(ajax-Write) := case
    next(state)=S2:TRUE;
    TRUE:FALSE;
esac;
next(ajax-Next) := case
next(state)=S3|next(state)=S4|next(state)=S5:TRUE;
    TRUE:FALSE;
esac;
......
```

图 7　CPMISS 系统用户信函模块 SMV 程序

将 SMV 程序和陷阱性质在模型检验器 SMV 中执行，分别可以得到与系统的可达性、活性、安全性相关的反例迹，并可以从中提取测试用例，如表 1、表 2 及表 3 所列。其中，P12 性质是验证网页 page-MailCenter 内部存在从 MailCenter9 子状态到 MailCenter0 子状态的迁移，根据设计的陷阱性质可以得到反例迹〈S0，S2，S9，S11，S2〉，通过反例抽取可以得到相应的测试用例(S0，link-MailCenter，S2，ajax-Write，S9，ajax-Send，S11，ajax-Home，S2)。也就是说，在 CPMISS 系统初始状态下，点击 page-Home(S0)页中的超链接 link-MailCenter 进入 page-MailCenter 页(S2)。下一步，点击“写邮件”按钮，系统发送异步请求 ajax-Write，在网页 page-MailCenter 中局部刷新出现写邮件的界面，此时，网页 page-MailCenter 进入子状态 MailCenter7(S9)。下一步，完成邮件的写作后，点击“发送邮件”按钮，系统发送异步请求 ajax-Send 发送邮件，发送成功后，网页 page-MailCenter 局部刷新显示发送邮件成功后的反馈界面，此时网页处于子状态 MailCenter9(S11)。下一步，点击“信函主页”按钮，系统发送异步请求 ajax-Home，局部刷新网页 page-MailCenter 的显示，使它又回到 MailCenter0 子状态(S2)。通过该测试用例的执行就可以测试在 CPMISS 系统的网页 page-MailCenter 内是否存在从状态 MailCenter9 到 MailCenter0 的迁移。

表 1　可达性测试用例生成

No.	State	Property formula	Counterexample	Test Case
P1	page-Login	AG¬page-Login	〈S0，S1〉	(S0，link，S1)
P2	page-UserCenter	AG¬page-UserCenter	〈S0，S2，S12〉	(S0，link，S2，link，S12)
P3	MailCenter6	AG¬MailCenter6	〈S0，S2，S3，S4，S6，S8〉	(S0，link，S2，ajax，S3，ajax，S4，ajax，S6，ajax，S8)
P4	MailCenter7	AG¬MailCenter7	〈S0，S2，S3，S5，S7〉	(S0，link，S2，ajax，S3，ajax，S5，ajax，S7)
P5	MailCenter10	AG¬MailCenter10	〈S0，S2，S9，S10〉	(S0，link，S2，ajax，S9，ajax，S10)
P6	MailCenter11	AG¬MailCenter11	〈S0，S2，S9，S11〉	(S0，link，S2，ajax，S9，ajax，S11)
...	...	...	...	...

表 2　活性测试用例生成

No.	Transition	Property formula	Counterexample	Test Case
P7	(page-Home，link，MailCenter0)	AG(page-Home→¬(link-MailPage∧EX MailCenter0))	〈S0，S2〉	(S0，link，S2)
P8	(page-Login，call，page-Home)	AG(page-Login→¬(call-LoginCheck∧EXpage-Home))	〈S0，S1，S0〉	(S0，link，S1，call，S0)
P9	(MailCenter0，ajax，MailCenter1)	AG(MailCenter0→¬(ajax-Inbox∧EX MailCenter1))	〈S0，S2，S3〉	(S0，link，S2，ajax，S3)
P10	(MailCenter1，ajax，MailCenter2)	AG(MailCenter1→¬(ajax-Desc∧EX MailCenter2))	〈S0，S2，S3，S4〉	(S0，link，S2，ajax，S3，sjax，S4)
P11	(MailCenter3，ajax，MailCenter3)	AG(MailCenter3→¬(ajax-Next(∧X MailCenter3))	〈S0，S2，S3，S5，S5〉	(S0，link，S2，ajax，S3，ajax，S5，ajax，S5)
P12	(MailCenter9，ajax，MailCenter0)	AG(MailCenter0→¬(ajax-Home∧EX MailCenter0))	〈S0，S2，S9，S11，S2〉	(S0，link，S2，ajax，S9，ajax，S11，ajax，S2)
...	...	...	...	...

表 3　安全性测试用例生成

No.	State	Property formula	Counterexample	Test Case
P13	page-Home	AG(page-Home→¬EX¬(page-Login∨MailCenter0))	〈S0，S3〉	(S0，link，S3)
P14	MailCenter0	AG(MailCenter0→¬EX¬(page-UserCenter∨MailCenter1∨MailCenter7))	〈S0，S2，S5〉	(S0，link，S2，ajax，S5)
P15	MailCenter1	AG(MailCenter1→¬EX¬(MailCenter0∨MailCenter2∨MailCenter3))	〈S0，S2，S3，S6〉	(S0，link，S2，ajax，S3，ajax，S6)
P16	MailCenter2	AG(MailCenter2→¬EX¬(MailCenter0∨MailCenter1∨MailCenter4))	〈S0，S2，S3，S4，S8〉	(S0，link，S2，ajax，S3，ajax，S4，ajax，S8)
P17	MailCenter3	AG(MailCenter3→¬EX¬(MailCenter0∨MailCenter1∨MailCenter2∨MailCenter3∨MailCenter5))	〈S0，S2，S3，S5，S6〉	(S0，link，S2，ajax，S3，ajax，S5，ajax，S6)
...	...	...	...	...

(下转第 244 页)

参 考 文 献

[1] Powell W B. Approximate Dynamic Programming: Solving the Curses of Dimensionality[M]. New York: John Wiley & Sons, 2007: 92-99

[2] Turk M, Pentland A. Eigen faces for recognition[J]. Journal of Cognitive Neuroscience, 1991, 3(1): 71-86

[3] Bartlett M S, Lades H M, Sejnowski T J. Independent component representations for face recognition[C]// Proceedings of The International Society for Optical Engineering. San Jose, 1998, 2399(3): 528-539

[4] Lee D, Seung H S. Learning the parts of objects by non-negative matrix factorization[J]. Nature, 1999, 401(6755): 788-791

[5] Lee D, Seung H S. Algorithms for non-negative matrix factorization[C]// Processings of Advances in Neural Information Processing Systems. Vancouver, 2000: 556-562

[6] Hoyer P O. Non-negative sparse coding [C]// Processings of IEEE Workshop on Neural Network for Signal Processing, Martigny. 2002: 557-565

[7] Hoyer P O. Non-negative matrix factorization with sparseness constrains[J]. Journal of Machine Learning Research, 2004, 5(9): 1457-1469

[8] Bucak S S, Günsel B. Incremental subspace learning via non-negative matrix factorization[J]. Pattern Recognition, 2009, 42(5): 788-797

[9] Guan Nai-yang, Tao Da-cheng, Luo Zhi-gang, et al. Online non-negative matrix factorization with robust stochastic approximation[J]. IEEE Transactions on Neural Networks and Learning Systems, 2012, 23(7): 1087-1099

[10] 姜伟,李宏,余震国,等. 稀疏约束图正则非负矩阵分解[J]. 计算机科学,2013,40(1):218-220,256

[11] 史加荣,焦李成,尚凡华. 不完全非负矩阵分解的加速算法[J]. 电子学报,2011,39(2):291-295

[12] 同鸣,张伟,张建龙,等. 一种基于部分基矩阵稀疏约束非负矩阵分解的抵抗大强度剪切攻击视频水印构架[J]. 电子与信息学报,2012,34(8):1819-1826

[13] 施蓓琦,刘春,孙伟伟,等. 应用稀疏非负矩阵分解聚类实现高光谱影像波段的优化选择[J]. 测绘学报,2013,42(3):351-358

[14] 方蔚涛,马鹏,成正斌,等. 二维投影非负矩阵分解算法及其在人脸识别中的应用[J]. 自动化学报,2012,38(9):1503-1512

[15] Liu Xu, Liu Tiao-tiao, Bai Wen-wen, et al. Encoding of rat working memory by power of multi-channel local field potentials via sparse non-negative matrix factorization [J]. Neurosci Bull, 2013, 29(3): 279-286

[16] Guan Nai-yang, Zhang Xiang, Luo Zhi-gang, et al. Sparse representation based discriminative canonical correlation analysis for face recognition[C]// Processings of International Conference on Machine Learning and Applications. Boca Raton, 2012: 51-56

(上接第 223 页)

结束语 Ajax 技术在当前主流的 Web 应用中被频繁地使用到。本文在对基于 Ajax 技术的 Web 应用的建模过程中,提出将网页中会引起局部刷新的部分划分成多个子部分即网页状态元素,定义网页状态元素上的 Ajax 操作集及相应的值域并根据这些网页状态元素的变化定义网页不同的子状态。将网页内部的所有状态封装在一起,使之与外部状态变化独立,这样能更好地反映网页内部的状态的变化,使得模型的结构更加清晰,方便针对一张网页内的状态迁移进行测试。本文利用 UML 状态图支持组合状态的特性来描述 Web 应用的模型,并将其转化为 Kripke 结构,使用 CTL 公式来描述模型的陷阱性质。使用模型检验工具 SMV 产生反例并抽取测试用例。在复杂的 Web 应用中一张网页内会出现异步刷新的部分并且 Ajax 请求通常会有很多,一张网页的内部结构和状态迁移会复杂很多,建模的难度非常大,下一步将研究如何简化对复杂的网页结构和状态迁移的 Web 应用建模。

参 考 文 献

[1] Garrett J J. Ajax: A New Approach to Web Applications [OL]. http://adaptivepath.com/ideas/essays/archives/000385.php

[2] Li L P, Miao H K, Chen S B. Test Generation for Web Applications Using Model-Checking[C]// SNPD. 2010: 237-242

[3] Grumberg, Long D E. Model checking and modular verification [J]. ACM Transactions on Programming Languages and Systems, 1994, 16(3): 843-871

[4] 颜炯,王戟,陈火旺. 基于模型的软件测试综述[J]. 计算机科学,2004,31(2):184-187

[5] 曾红卫. Web 应用的验证与测试方法研究[D]. 上海:上海大学,2008

[6] Hierons R M, Bogdanov K, Bowen J P, et al. Using formal specifications to support testing [J]. ACM Computing Surveys (CSUR), 2009, 41(2): 1-76

[7] Ammann P, Ding W, Xu D. Using a model checker to test safety properties[C]// Proceedings of the 7th International Conference on Engineering of Complex Computer Systems (ICECCS 2001). IEEE Press: New York, 2001: 212-221

[8] Fraser G, Wotawa F, Ammann P E. Testing with model checkers: a survey[J]. Software Testing, Verification and Reliability, 2009, 19(3): 215-261

[9] Marchetto A, Tonella P, Ricca F. State-based testing of Ajax Web applications[C]// Proceedings of the 1st IEEE International Conference on Software Testing Verification and Validation (ICST'08). IEEE Computer Society, 2008: 120-130

[10] Mesbah A, Bozdag E, van Deursen A. Crawling Ajax by Inferring User Interface State Changes [C]// Eighth Interational Conference: Web Engineering, ICWE'08. 2008: 122-134

[11] Mesbah A, van Deursen A, Roest D. Invariant-based automatic testing of modern Web applications[J]. IEEE Transactions on Software Engineering (TSE), 2012, 38(1): 35-53

[12] McMillan K L. The SMV System for SMV version 2.5.4 [OL]. http://www.cs.cmu.edu/modelcheck/smv/smvmanual.ps

[13] Fraser G, Gargantini A. An evaluation of model checkers for specification based test case generation[C]// ICST. IEEE Computer Society, 2009: 41-50