

基于 Object-Z 与 Markov 链的校园卡系统测试用例

仲晓敏¹, 侯建花², 杨长青²

(1 淮海工学院 计算机工程学院, 江苏 连云港 222005; 2 淮海工学院 理学院, 江苏 连云港 222005)

摘 要: 针对提高校园卡系统准确性和可靠性测试的要求, 提出了 Object-Z 与 Markov 链结合的测试用例自动生成算法. 使用 Object-Z 对系统进行形式化规约, 生成测试场景和操作顺序图; 将操作顺序图转换为 Markov 链使用模型; 根据测试场景和 Markov 链使用模型生成数量相对合理的测试用例. 该方法无需对系统进行运行, 在需求分析与测试阶段就能对系统的功能进行测试. 生成的校园卡系统测试用例证明该方法是有用的, 并且在提高测试覆盖率的同时, 使用 Markov 链也能保证对系统的可靠性测试.

关键词: 校园卡系统; Object-Z; 测试场景; Markov 链; 使用模型; 测试用例

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2012)03-0073-05

Test Case Generator for Campus-wide Card System Based on Object-Z and Markov

ZHONG Xiao-min¹, HOU Jian-hua², YANG Chang-qing²

(1 School of Computer Engineering, Huaihai Institute of Technology, Lianyungang 222005, China;

2 School of Mathematics & Physics, Huaihai Institute of Technology, Lianyungang 222005, China)

Abstract: For improving the accuracy and reliability of the Campus card system test requirement, test cases generator based on Object-Z and Markov chain is proposed in this paper. The formal specification of the system is gained by using Object-Z, then test scenarios and operation sequence diagrams are generated. Operation sequence diagrams are converted into a Markov chain usage model. According to the test scenarios and the Markov chain model to produce a relatively reasonable amount of test cases. The method don't need to run the system, and could be able to test the functionality of the system in the time of requirements analysis. The results of generated test cases for campus-wide card system show the method is effective, and using the Markov chain ensures the reliability of the system test.

Key words: campus-wide card; Object-Z; test scenario; Markov chain; usage model; test case

1 引言

校园卡系统以符合国际标准 ISO/IEC14443 的 IC 卡替代传统饭卡、学生证、借书证、考试证等各类校内证件. 使用面向对象方法开发的校园卡系统, 基本可测试单元是类或对象. 类或对象具有封装、继承、多态等三个特性. 封装隐藏了类的数据结构和实现细节, 不易进行白盒测试; 继承允许子类复用父类, 出现重复测试; 多态和动态绑定引入不可判定问

题, 很难静态地确定在给定的测试用例中哪个方法被激活.

Object-Z 是形式化语言 Z 的面向对象扩充^[1], 基于集合论与数理逻辑, 具有精确的描述性和很强的抽象性, 为面向对象软件的测试用例生成提供语义上的支持. 将校园卡系统以 Object-Z 的模式进行描述, 能够在一个较高的抽象层上对功能进行概括, 满足对功能测试的要求, 而且测试用例从规约得出, 与程序结构无关. Markov 链以数学模型的方法表

收稿日期: 2011-07-20; 修回日期: 2011-08-18

基金项目: 国家自然科学基金项目(50806011)

示软件的使用特征,构建起使用模型^[2]. 基于软件的使用模型产生软件的随机测试用例,可以保证对软件可靠性的测试^[3].

目前基于 Object-Z 测试用例生成的研究^[4-5]都只是将 Object-Z 形式规格说明转换成有限状态机,生成的测试用例仍然存在状态爆炸等问题;也很难保证测试用例的可靠性. 而基于 Markov 链的测试用例生成^[6-7]在系统状态数较大时,容易出现拉普拉斯反演的困难^[8]. 本文就结合 Object-Z 和 Markov 链的校园卡系统测试用例生成进行研究.

2 校园卡系统的 Object-Z 形式化规约

2.1 Object-Z 形式化规约方法

Object-Z 通过类的概念来封装对象的状态及相关操作,并与面向对象程序设计语言中的类概念相对应. Object-Z 类包括状态模式和若千操作模式,操作模式描述类属性与输入/输出变量之间的关系;操作模式与状态模式的初始值一起决定了类对象所有可达的状态. 类模式的语法结构如图 1 所示^[9].

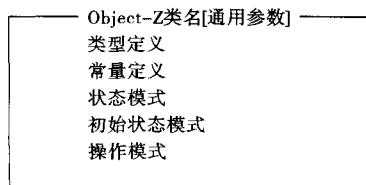


图 1 Object-Z 类模式的语法结构

基于面向对象的思想, Object-Z 类之间能够相互调用,类可以捕捉一组个体对象的行为、关系及交互. 此外,对于继承、多态和封装等面向对象的概念, Object-Z 都有很好的支持.

2.2 校园卡系统的 Object-Z 规约

校园卡系统定义三种对象来表示用户:教师、学生和临时使用者,并由 teacher、student 和 tempuser 类实例化而成. 基本类 tempuser 只具备消费权限; student 类继承 tempuser 类,拥有享受大多数校园服务的权限; teacher 类继承 student 类,拥有更高的校园服务享受权限,并且账户的有效期更长.

校园卡系统的 Object-Z 的规约包括对 Teacher、Student 和 Tempuser 三个类,以及类 CampusWideCardSystem 的 Object-Z 形式化规约, CampusWideCardSystem 类包括 Teacher, Student 和 Tempuser. 因为篇幅所限,只在图 2 中展示了 CampusWideCardSystem 类的形式化规约,其中 Consume 等六个操作定义了校园卡系统与外界

SelectAccount 以及 SelectStudentAccount 使用了一个多态类型的变量 (polymorphic type variable): account?. SelectAccount 可以选择任一种的账户,而 SelectStudentAccount 只能选择 Teacher 或是 Student 账户.

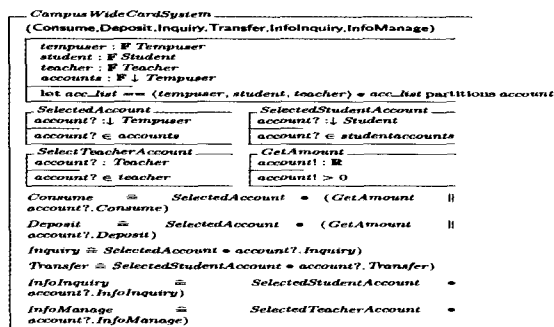


图 2 CampusWideCardSystem 类的形式化规约

3 校园卡系统的测试场景生成

3.1 系统的顶层操作图

测试场景是与待测软件的某次执行相对应的一个活动场景^[10]. 分析待测软件的执行所包括的各个操作,生成操作顺序. 其中顶层操作图可以从 Object-Z 形式化规约推导得到. 校园卡系统的顶层操作图如图 3,节点 Pn1 为等待的操作选择,图中列举了校园卡系统 Object-Z 规约定义的六个操作,这些操作可以被重复执行,也可以在执行完某一个操作后退出系统.

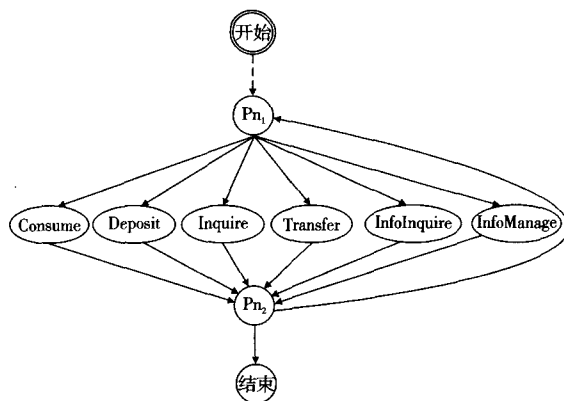


图 3 校园卡系统的顶层操作图

校园卡系统的规约定义的六个操作之间的关系,有如下定义:

CWC $\triangle$ Deposit[] Consume[] Inquiry[]
Transfer[] infoManage[] infoInquiry

上式说明系统六个顶层操作的调用可以是独立的.

3.2 操作调用顺序规则的定义

在获得校园卡系统的顶层操作图后,对校园卡系统 Object-Z 规约中的每个操作进行剖析. Object-Z 规约中存在两种类型的操作:元操作和组合操作. 元操作没有对其它操作的调用过程,它的调用顺序永远为空,记为 $\langle \text{nil} \rangle$. 组合操作包括合取($\wedge$)、并行($\parallel$)、顺序($;$)和选择($[\]$)等四种. 下面给出这四种组合操作调用顺序规则的定义.

(1)合取操作($\wedge$):被组合的各操作同时执行,各个子操作是并发调用的,调用顺序定义如下:

$$\begin{aligned} \text{Op} &\triangleq \text{Op}_1 \wedge \text{Op}_2 &<\text{Op}_1, \text{Op}_2> \\ &&<\text{Op}_2, \text{Op}_1> \\ \text{Op} &\triangleq \text{Op}_1 \wedge \text{Op}_2 \wedge \text{Op}_3 &<\text{Op}_1, \text{Op}_2, \text{Op}_3> \\ &&<\text{Op}_2, \text{Op}_1, \text{Op}_3> \\ &&<\text{Op}_3, \text{Op}_1, \text{Op}_2> \\ &&<\text{Op}_3, \text{Op}_2, \text{Op}_1> \end{aligned}$$

(2)并行操作($\parallel$):组合不同的并发操作,完成相互之间的通讯. 调用操作顺序定义如下:

$$\begin{aligned} \text{Op} &\triangleq \text{Op}_1 \parallel \text{Op}_2 &<\text{Op}_1, \text{Op}_2> \\ \text{Op} &\triangleq \text{Op}_1 \parallel \text{Op}_2 \parallel \text{Op}_3 &<\text{Op}_1, \text{Op}_2, \text{Op}_3> \end{aligned}$$

譬如,校园卡系统的一个操作 $\text{Deposit} \triangleq \text{SelectAccount} \cdot (\text{GetAmount} \parallel \text{account!} \cdot \text{Deposit})$ 中,首先激活 SelectAccount ,然后激活 GetAmount 去获取输出变量 amt! ,并将其作为输入变量传递给 $\text{account!} \cdot \text{Deposit}$;因此,调用顺序为 $\text{SelectAccount}, \text{GetAmount}$,然后才是 $\text{account!} \cdot \text{Deposit}$.

(3)顺序操作($;$):将按照一定顺序依次执行的各个子操作组合起来. 提供了与并行操作不同的通讯方式. 因此调用顺序定义如下:

$$\begin{aligned} \text{Op} &\triangleq \text{Op}_1; \text{Op}_2 &<\text{Op}_1, \text{Op}_2> \\ \text{Op} &\triangleq \text{Op}_1; \text{Op}_2; \text{Op}_3 &<\text{Op}_1, \text{Op}_2, \text{Op}_3> \end{aligned}$$

(4)选择操作($[\]$):根据情况在被组合的子操作中选择其中一个操作来执行. 因此调用顺序定义如下:

$$\begin{aligned} \text{Op} &\triangleq \text{Op}_1 [\] \text{Op}_2 &<\text{nil}> \\ &&<\text{Op}_1> \\ &&<\text{Op}_2> \end{aligned}$$

若将合取、并行、顺序、选择等四种操作组合起来,调用顺序依优先级及结合律来决定,它们的优先级顺序是:合取和并行,顺序、选择. 对于多态对象声明的调用顺序,进行如下定义:

$$\text{Op} \triangleq \text{SelectOne} \cdot a! \cdot \text{Op}_a; ?$$

$$\begin{aligned} &<\text{SelectOne}, a! \cdot (\text{A}), \text{Op}_a> \\ &<\text{SelectOne}, a! \cdot (\text{B}), \text{Op}_a> \\ &<\text{SelectOne}, a! \cdot (\text{C}), \text{Op}_a> \end{aligned}$$

其中假设 A、B 和 C 是三个已经定义了的类. 类 B 继承类 A,而类 C 继承类 B. 多态对象 $a!; ?$ 申明在 SelectOne 操作中. Op_a 是类 A 的一个操作,且被类 B 及 C 所继承.

3.3 操作调用顺序图

为每个操作推导出操作调用图(operation invocation diagram, OID). 每个操作的每个可能调用顺序组成 OID 的一个可能路径. 例如:上述多态对象 $a!$ 的 Op 操作将会生成三个可能路径. 在生成了 OID 之后即藉以推导出分层调用顺序图(hierarchical invocation diagram, HID). 仅以 Consume 操作为例,图 4 为它的 HID.

$\langle \text{SelectAccount}, \text{GetAmount}, \text{account!} \cdot (\text{Teacher}). \text{Withdrawal} \rangle$
 $\langle \text{SelectAccount}, \text{GetAmount}, \text{account!} \cdot (\text{Student}). \text{Withdrawal} \rangle$
 $\langle \text{SelectAccount}, \text{GetAmount}, \text{account!} \cdot (\text{Tempuser}). \text{Withdrawal} \rangle$

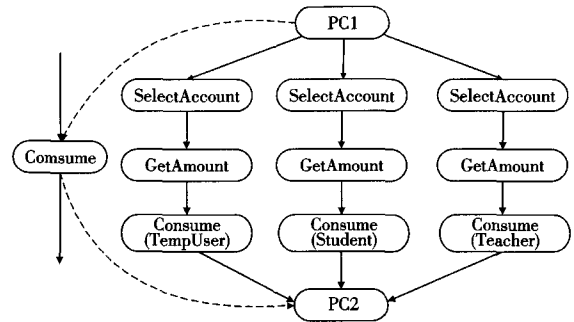


图4 校园卡系统 Consume 操作的 HID

3.4 测试场景的生成

在得到待测软件的 HID 后, HID 的每个可能路径即为一个可能的测试场景. 以下列出校园卡系统的两个可能测试场景作为示例:

(1) ($\langle \text{Consume} \langle \text{SelectAccount}, \text{GetAmount}, \text{account?} \cdot (\text{Tempuser}). \text{Consume} \rangle \rangle$)

(2) ($\langle \text{Deposit} \langle \text{SelectAccount}, \text{GetAmount}, \text{account?} \cdot (\text{Teacher}). \text{Deposit} \rangle, \text{Inquiry} \langle \text{SelectAccount}, \text{account?} \cdot (\text{Teacher}). \text{Inquiry} \rangle \rangle$)

4 校园卡系统的测试用例生成

4.1 系统操作的 Markov 链表示

Markov 可以在不同的粒度建立软件系统的使用模型. 在预期的运行环境、用户及运行模式下,校

园卡系统与外部运行环境及用户的交互是遵从一定概率分布的随机事件,因此可以根据校园卡系统的 Object-Z 规约对系统的运行情况建立 Markov 链使用模型。

Markov 使用模型的构建包括两个内容:模型结构的构造和转移概率的确定。模型结构由系统运行过程中的状态节点构成;设状态集为 $S = \{s_i | i \in \{1, 2, \dots, n\}\}$, 其中 n 为状态的总数;弧表示状态间的转移关系,每条弧都有一个转移概率。转移概率标志状态转移发生的可能性,设状态转移矩阵为 $p = \{p(i, j) | i, j \in \{1, 2, \dots, n\}\}$, 其中 $p(i, j)$ 为状态 i 向状态 j 转移的概率,且 $0 \leq p(i, j) \leq 1$, $\sum_j p(i, j) = 1$, 即所有弧的转移概率之和应该为 1。

将第 2 节得到的校园卡系统顶层操作图, HID 图转化为 Markov 使用模型。在已经构造出的测试场景下,将输入的顶层操作图和 HID 进行综合和化

简,然后对各状态节点进行遍历,若某状态节点为分支节点,则对该节点调用分枝遍历算法。最终得到 $\{OP_i | OP_i = \langle O_i, P_i \rangle, i = 1, 2, \dots, N\}$, 其中 O_i 表示一个操作调用, P_i 表示这个操作调用发生的概率;若以有向图表示更为直观。同样以 Consume 操作为例,得到的 Markov 链使用模型如图 5 所示。

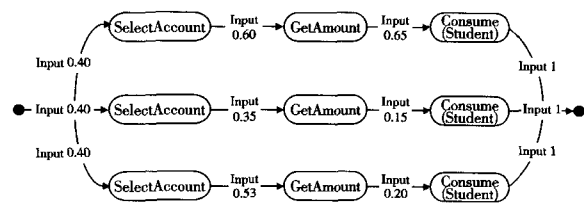


图 5 校园卡系统 Consume 操作的 Markov 链使用模型

4.2 测试用例的生成

测试用例的核心是输入数据,表 1 是校园卡系统的输入/输出数据。

表 1 校园卡系统的输入/输出数据

In/Out	名称	类名	类型	范围	分布
In	amount!	CompusWideCardSystem	Real	$0 \leq \text{amount!} \leq 300$	equal
In	Amount?	CompusWideCardSystem	Tempuser	$00001 \leq \text{amount?} \leq 10000$	
In	Amount?	CompusWideCardSystem	Student	$10001 \leq \text{amount?} \leq 20000$	
In	Amount?	CompusWideCardSystem	Teacher	$20001 \leq \text{amount?} \leq 30000$	
Out	get!	CompusWideCardSystem	Real	≥ 0	

Markov 链使用模型中启动状态到停止状态的随机路径上,状态节点和各弧的概率值构成序列可得到一个测试用例。以下面的测试场景为例:

($\langle \text{Consume} \langle \text{SelectAccount}, \text{GetAmount}, \text{account?} (\text{Teacher}), \text{Consume} \rangle \rangle$)

由图 5 得知转移概率为 0.20,设需生成测试用例数为 1000,其中该测试场景为 20 个。据表 1, amount! 的分布型态定义为 equal,所以均匀的从 0 到 300 间随机得到。以下是所生成的两个测试用例,有 # 的是账户号码, \$ 开始的则代表金额。

Test case for test scenario 1: ($\# 00506, \$ 100.00$)

Test case for test scenario 2: ($\# 22301, \$ 2000.00$)

5 结束语

文中提出了用 Object-Z 语言对校园卡系统进行规约,并根据相应的规约得到操作调用顺序图和

测试场景,然后按照一定的方法将操作顺序图转换为 Markov 链使用模式。最后联合测试场景和 Markov 链使用模式生成校园卡系统的测试用例。以上方法有着以下的优点:能自动生成测试用例,而且根据 Markov 链使用模型的转移概率生成数量合理的测试用例。从而提高了测试效率,这种方法不仅适合于软件的系统测试,同时也适用于软件设计阶段对软件需求和设计模型的测试和验证。

参考文献:

- [1] 胡燕京,张毅坤,朱伟,等. 嵌入式软件功能路径测试用例自动生成研究[J]. 微电子学与计算机, 2010, 27(1): 81-85.
- [2] Nalini R, Liu Zhao-hui, Bonnie K R. NHPP models with Markov switching for software reliability[J]. Computational Statistics & Data Analysis, 2008, 52(8): 3988-3999.
- [3] 张旭,石柱,王崑声. 基于使用剖面的软件可靠性测试用例生成方法[J]. 计算机仿真, 2009, 26(12): 66-69.

- [4] 朱义毅, 缪淮扣, 唐新香. 一种基于 Object-Z 规格说明的集成测试用例生成方法[J]. 上海大学学报: 自然科学版, 2009, 15(1): 60-65.
- [5] Ashraf A, Nadeem A. Automating the generation of test cases from Object-Z specifications[C]// 30th Annual International Computer Software and Applications Conference (COMPSAC '06). Chicago, Compsac, 2006: 101-104.
- [6] Vilkomir S A, Swain W T, Poore J H. Combinatorial test case selection with markovian usage models[C]// Fifth International Conference on Information Technology: New Generations (itng 2008). Las Vegas, IEEE Computer Society, 2008: 3-8.
- [7] Prowell S J. Computations for markov chain usage models[R]. Tennessee: The University of Tennessee UT-CS-03-505, 2003.
- [8] 于敏, 何正友, 钱清泉. 基于 Markov 过程的硬/软件综合系统可靠性分析[J]. 电子学报, 2010, 38(2): 473-479.
- [9] 缪淮扣, 李刚, 朱关铭. 软件工程语言——Z[M]. 上海: 上海科学技术文献出版社, 2000.
- [10] Sapna P G, Mohanty H. Automated test scenario selection based on levenshtein distance[C]// The 6th International Conference on Distributed Computing and Internet Technologies (ICDCIT 2010). Bhubaneswar, Springer Berlin/Heidelberg, 2010: 255-266.

作者简介:



仲晓敏 女, (1977—), 硕士, 讲师. 研究方向为软件工程、人工智能.

(上接第 72 页)

作者简介:



李 慧 女, (1979—), 硕士研究生, 讲师. 研究方向为数据挖掘、智能信息处理.

胡 云 女, (1978—), 硕士研究生, 讲师. 研究方向为人工智能.

李存华 男, (1963—), 博士研究生, 研究员, 教授. 研究方向为数据挖掘、人工智能.