软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2011,22(7):1457−1474 [doi: 10.3724/SP.J.1001.2011.03872]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 确定性有限状态机的最小测试成本迁移覆盖准则*

刘 攀[1,2+], 缪淮扣[1,2], 曾红卫[1,2], 梅 佳[1]

[1](上海大学 计算机工程与科学学院,上海 200072)

[2](上海市计算机软件评测重点实验室,上海 201112)

## DFSM-Based Minimum Test Cost Transition Coverage Criterion

LIU Pan[1,2+], MIAO Huai-Kou[1,2], ZENG Hong-Wei[1,2], MEI Jia[1]

[1](School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)

[2](Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China)

+ Corresponding author: E-mail: liupan@shu.edu.cn, http://www.shu.edu.cn

**Liu P, Miao HK, Zeng HW, Mei J. DFSM-Based minimum test cost transition coverage criterion. *Journal of Software*, 2011,22(7):1457−1474.** http://www.jos.org.cn/1000-9825/3872.htm

**Abstract**: A key problem in software testing is having the right cost/benefit tradeoffs for the software that is being tested. Based on the 2-8 law of fault distribution in programs, the paper divides the test process in two stages for solving this problem. The first stage is to find the faults in software at minimum cost, and the second stage is to add some additional test cases for those faults found in the first stage to detect more potential faults in software. The paper lays emphasis on the realization of the first stage. According to the theories of both the deterministic finite state machine (DFSM) and set partition, a DFSM-based minimum test cost transition coverage criterion is presented in this paper, and the criterion's sufficiency and necessary conditions are given. Moreover, two algorithms that realize the optimal transition coverage and the minimum test cost transition coverage are designed, and the effectiveness of the test set is also discussed. In the experiments, the method not only obtains a set of test cases with the minimal size and the shortest total length of all test cases, but also finds all faults in transitions of the DFSM.

**Key words**: minimum test cost transition coverage criterion; cost/benefit tradeoff; DFSM (deterministic finite state machine); transition coverage

**摘 要**: 软件测试中的一个重要问题是测试成本和测试效率的平衡问题.依据程序中错误分布的 2-8 定律,将测试分为两个阶段,以解决该问题.第 1 阶段采用最小代价发现软件中的错误,第 2 阶段针对第 1 阶段中发现的错误补充设计测试用例,探测软件中潜在的错误.重点是第 1 阶段的实现.依据确定性有限状态机和集合划分的理论,提出了确定性有限状态机的最小测试成本迁移覆盖准则,给出了最小测试成本迁移覆盖存在的充分和必要条件,设计了优化迁移覆盖和最小测试成本迁移覆盖的实现算法,并讨论了测试序列集合的有效性问题.在实验中,依据该方法不仅能够获得最小测试成本的测试用例集合,而且同样能够探测出确定性有限状态机中迁移上的错误.

**关键词**: 最小测试成本迁移覆盖准则;成本/效率的平衡;确定性有限状态机;迁移覆盖

* 基金项目: 国家自然科学基金(60970007, 61073050); 国家重点基础研究发展计划(973)(2007CB310800); 上海市自然科学基金(09ZR1412100); 上海市科学技术委员会资助项目(10510704900); 上海市重点学科建设项目(J50103)

收稿时间: 2009-12-28; 修改时间: 2010-03-29; 定稿时间: 2010-04-28

中图法分类号: TP311 文献标识码: A

软件测试是发现软件中的错误、保证软件质量的一种重要手段.在软件开发过程中,软件测试成本已占软件开发总成本的 50%左右.因此,改进软件测试技术、降低软件测试成本显得尤为必要.基于模型的测试可由软件需求的规格说明自动生成测试用例,节约了测试时间和测试成本,因此,该方法受到业内人士的高度重视,成为软件测试领域的一个新的挑战[1].

有限状态机(finite state machine,简称 FSM)[2]是一种描述软件需求的形式规格说明描述工具,具有精确性、可推导性和可验证性等优点.因此,基于有限状态机的测试用例生成技术被广泛研究,形成了一系列的测试覆盖准则.Chow[3]提出 W 方法,将 FSM 转换成一棵测试生成树,遍历测试生成树,得到由根节点到所有叶子节点的测试序列.Huang[4]提出了基于 FSM 的分枝覆盖,其本质是遍历 FSM 中的每一条迁移.Howden[5]提出了内部边界覆盖,对 FSM 中所有状态覆盖一次.Pimont 和 Rault[6]提出了开关覆盖,该方法得到的测试用例集合与迁移对覆盖[7,8]得到的测试用例集合完全一致,但两者均未考虑到多个迁移对之间的组合.为此,Fujiwara 等人[9]在 Pimont 研究的基础上提出了 $n$ 迁移对覆盖.另外,还有随机遍历 FSM 产生测试用例的 T 方法[10]和用于网络协议测试的 U 方法[10,11]等.

测试覆盖准则的目的是发现一个最小的有效测试用例集合,测试软件的每一部分[7].实际测试时,无法穷尽测试软件的每一部分,因此必须在测试成本和测试效率之间加以权衡(tradeoff)[8].影响测试成本和测试效率的因素有很多,本文仅考虑测试用例的数目、测试用例的总长度及测试用例集合的覆盖能力对测试成本和测试效率的影响.由于测试意图不同,现有的基于 FSM 模型的测试覆盖准则在测试成本和测试效率之间的取舍也不相同.状态覆盖准则仅需覆盖 FSM 中的每一状态,因此测试成本较低.但是,由于存在未被覆盖的迁移,因此状态覆盖准则的测试效率较低.迁移覆盖准则需要覆盖 FSM 中的每条迁移,具有相当强的错误探测能力.然而迁移覆盖准则本身并没有直接给出实现方法,常用的实现方法有欧拉图和迁移旅行等,但欧拉图的应用条件过于苛刻,而迁移旅行生成的迁移序列可能包含冗余迁移.W 方法给出了测试生成的实现方法,然而生成的测试用例集合中测试用例数目过多且存在大量的冗余迁移[12],导致软件测试成本有所增加.在实际软件测试中,80%的错误来源于 20%的代码.若测试用例集合满足迁移对覆盖或 $n$ 迁移对覆盖(测试效率较高的测试覆盖准则),其中大多数测试用例并不能发现任何错误,而其中几条测试用例发现的错误往往位于同一个模块当中,造成测试成本的极大浪费.

为此,本文提议将测试划分为两个阶段,解决测试成本和测试效率的平衡问题:

- 第 1 阶段产生覆盖 FSM 中所有迁移的最小测试成本的测试用例集合(完备的、无冗余或冗余较少、测试用例数目少、测试用例总长度最短),并测试软件;
- 第 2 阶段针对测试第 1 阶段中发现的错误进行重点测试.

第 1 阶段用最小成本测试 FSM 中的每一条迁移,第 2 阶段针对发现错误的位置增加测试用例,探测软件中潜在的错误.本文的目的是设计最小测试成本迁移覆盖准则,实现第 1 个阶段.该准则的实现思想是:在 FSM 迁移覆盖的基础上,找到迁移集合 $T$ 的一个有效划分 $\pi$,使得构成 $\pi$ 的所有非空子集均为一条有效的测试序列,且这些非空子集的数目最少.最小测试成本迁移覆盖准则产生的是迁移序列集合,迁移序列是测试用例具体化的必要工作,它确定了测试用例的范围,对测试用例的生成具有决定性的指导作用.

本文的贡献:

1) 建议将测试分为两个阶段来获取测试成本和测试效率之间的平衡;
2) 提出了最小测试成本迁移覆盖准则.依据该准则生成的测试用例集合覆盖了 FSM 中的所有迁移,同时实现了测试成本最小化.因此,该方法可缓解基于模型的测试时可能存在的状态组合爆炸问题[13];
3) 设计了最小测试成本迁移覆盖的实现算法.

文中涉及到 FSM 的定义(定义 1)、迁移和迁移序列的描述,序列的连接定义(定义 6)来自于文献[2,14−16],迁移覆盖(定义 3)和最小测试成本迁移覆盖(定义 8)借鉴了集合的覆盖和划分的定义.为适用需求,本文统一采

用形式化方法对相关定义进行描述.在此基础上,本文给出了最小测试成本迁移覆盖准则的相关定义和定理.

本文第 1 节通过一个案例演示测试两个阶段的解决方案.第 2 节提出迁移序列覆盖准则.第 3 节给出最优迁移序列覆盖存在的必要性定理,设计优化迁移序列覆盖的实现算法,讨论迁移序列的有效性.第 4 节给出最优迁移序列覆盖存在的充分性定理,设计最优迁移序列覆盖的实现算法.第 5 节将本文的研究与几种基于 FSM 的测试覆盖准则进行实验对比.第 6 节分析国内同类研究.第 7 节总结全文,并指出今后的研究方向.

## 1 动 机

通过一个实例,本文演示了将测试分为两个阶段、平衡测试成本和测试效率的解决方案.假定一个 FSM 如图 1 所示,其中,节点 1~节点 11 表示 FSM 的状态,边上的条件表示 FSM 中的迁移符号.该 FSM 对应程序 $A$,FSM 中的符号 $\tau$表示程序 $A$ 的内部动作,程序 $A'$为程序 $A$ 的错误程序,其中包含了 5 个错误.本例的目的是由 FSM 导出测试路径和测试用例,探测错误程序 $A'$中的 5 个错误.程序 $A$、错误程序 $A'$及错误代码如下:

| 程序 $A$ | | 错误程序 $A'$ | | 错误代码 |
|---|---|---|---|---|
| 1 | read ($a$,$b$,$c$,$d$); | 1 | read ($a$,$b$,$c$,$d$); | |
| 2 | if ($a$>0 && $b$>0 && $c$>0) | 2 | if ($a\geqslant$0 && $b\geqslant$0\|\|$c$>0) | error 1: $a\geqslant$0, error 2: $b\geqslant$0, error 3: \|\| |
| 3 | if ($c$>2) | 3 | if ($c\geqslant$2) | error 4: $c\geqslant$2 |
| 4 | output (2); | 4 | output (2); | |
| | else | | else | |
| 5 | output (4); | 5 | output (4); | |
| 6 | endif | 6 | endif | |
| 7 | output (6); | 7 | output (6); | |
| 8 | endif | 8 | endif | |
| 9 | if ($d$>0) | 9 | if ($d\geqslant$0) | error 5: $d\geqslant$0 |
| 10 | output (8); | 10 | output (8); | |
| 11 | endif | 11 | endif | |

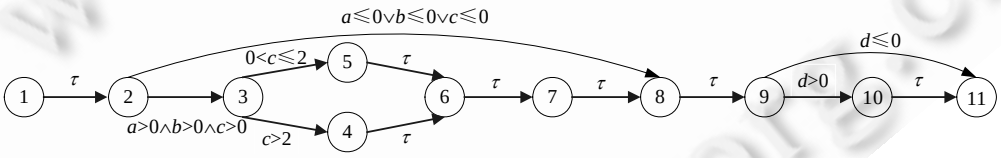



Fig.1 A FSM

图 1 一个 FSM

依据路径覆盖、迁移覆盖和状态覆盖,由 FSM 分别得到一组测试路径,并为这些测试路径设计相应的测试用例,再用这些测试用例探测程序 $A'$中的错误.具体的测试路径、对应的测试用例及探测到的错误如下:

| 测试路径: | 测试用例: | 揭示程序 $A'$中的错误: |
|---|---|---|
| $P_1$=1-2-8-9-11; | $tc_1$=($a$=0,$b$=1,$c$=0,$d$=0) | error 1, error 5 |
| $P_2$=1-2-8-9-10-11; | $tc_2$=($a$=0,$b$=0,$c$=0,$d$=1) | error 1, error 2 |
| $P_3$=1-2-3-4-6-7-8-9-11; | $tc_3$=($a$=1,$b$=1,$c$=2,$d$=−1) | error 4 |
| $P_4$=1-2-3-4-6-7-8-9-10-11; | $tc_4$=($a$=2,$b$=2,$c$=2,$d$=3) | error 4 |
| $P_5$=1-2-3-5-6-7-8-9-11; | $tc_5$=($a$=3,$b$=1,$c$=2,$d$=−2) | error 4 |
| $P_6$=1-2-3-5-6-7-8-9-10-11; | $tc_6$=($a$=4,$b$=4,$c$=2,$d$=2) | error 4 |

满足路径覆盖的测试路径为 $P_1$~$P_6$,满足迁移覆盖的测试路径为 $P_1$,$P_4$ 和 $P_6$,满足状态覆盖的测试路径为 $P_4$ 和 $P_6$.由测试用例覆盖的路径、测试用例及其揭示的错误可知:

1) 满足路径覆盖的测试用例 $tc_1$~$tc_6$ 揭示错误 error 1,error 2,error 4 和 error 5,满足迁移覆盖的测试用例 $tc_1$,$tc_4$ 和 $tc_6$ 揭示错误 error 1,error 4 和 error 5,满足状态覆盖的测试用例 $tc_4$ 和 $tc_6$ 揭示错误 error 4;
2) 即使是覆盖能力很强的路径覆盖仍然不能发现程序 $A'$中的所有错误,如 error 3;
3) 对比测试成本(测试路径的数目)和错误探测能力,迁移覆盖成本是路径覆盖成本的 50%,同时,迁移覆盖的错误探测能力是路径覆盖错误探测能力的 75%;
4) 状态覆盖成本最低,但其错误探测能力仅为路径覆盖的 25%.

为平衡测试成本和测试效率,本文建议采用两个阶段测试程序 $A'$.第 1 阶段采用迁移覆盖发现程序 $A'$中的错误 error 1,error 4 和 error 5.第 2 阶段针对 FSM 中迁移出错的位置,结合边界值分析的方法,补充设计一些测试用例,以期待发现程序 $A'$中的更多错误.补充测试用例的设计如下:

| 已发现的错误 | FSM 中的出错状态 | 补充测试用例 |
|---|---|---|
| error 1 | if ($a$>0 && $b$>0 && $c$>0) | $tc'_7$ =($a$=1,$b$=0,$c$=1), $tc'_8$ =($a$=1,$b$=1,$c$=0), $tc'_9$ =($a$=0,$b$=1,$c$=1) |
| error 4 | if ($c$>2) | $c$=3(由于 $c$=1 和 $c$=2 已出现在测试用例 $tc'_7$ 和 $tc_4$ 中) |
| error 5 | if ($d$>0) | $d$=−1(由于 $d$=0,$d$=2 已出现在测试用例 $tc_1$ 和 $tc_6$ 中) |

综合补充测试用例中变量的取值,最终设计的测试用例为 $tc_7$=($a$=1,$b$=0,$c$=1,$d$=−1),$tc_8$=($a$=1,$b$=1,$c$=0,$d$=−1), $tc_9$=($a$=0,$b$=1,$c$=1,$d$=−1),$tc_{10}$=($a$=1,$b$=0,$c$=3,$d$=−1).其中,$tc_7$ 和 $tc_{10}$ 发现了程序 $A'$中的错误 error 2,$tc_8$ 和 $tc_9$ 发现了程序 $A'$中的错误 error 3.第 2 阶段后,程序 $A'$中所有的错误均被发现.本文的这种测试思想充分应用了程序中错误分布的 2-8 定律.若在未知程序错误的情况下盲目增加测试用例,不但会大量增加测试成本,而且增加的测试用例不一定能够发现程序中的错误.若采用迁移覆盖产生测试用例,并测试程序,再针对发现的错误补充设计少量测试用例,将能更有效地探测出程序中潜在的错误.为了降低第 1 阶段测试中迁移覆盖的成本,本文研究了最小测试成本迁移覆盖准则.

## 2 迁移覆盖

确定性有限状态机(deterministic finite state machine,简称 DFSM)是一种能够实现状态转移的自动机.对于一个给定的属于该自动机的状态和一个属于该自动机字母表内的字符,它都能根据事先给定的转移函数转移到下一个确定的状态.

**定义 1(DFSM)**. 确定性有限状态机为五元组 $M=\langle S,I,s_0,T,f\rangle$,其中:

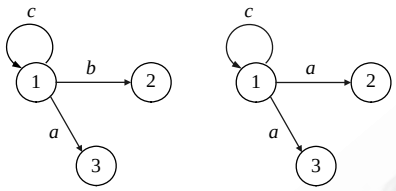
- $S$ 为非空有限状态集合;
- $I$ 为非空有限输入字符集合;
- $s_0\in S$ 为初始状态;
- $T$ 为终止(可接受)状态集合,$T\subseteq S$;
- $f$ 为迁移函数 $f:S\times I\to S$.

由定义 1 可知,一个 DFSM 是一个状态转换图.与普通的图不同,在 DFSM 中,状态的迁移是唯一的.例如,图 2(a)是一个 DFSM,而图 2(b)则不是一个 DFSM,因为在图 2(b)中,当状态 1 迁移到状态 2 和状态 3 时,输入字符均为 $a$,因此图 2(b)是非确定有穷自动机(nondeterministic finite state machine,简称 NDFSM).目前,绝大多数软件的需求都能用 DFSM 建模.因此,本文所指的 FSM 是特指 DFSM(简称 $M$).定义 1 中的符号及其含义在本文中可以直接使用.

在 FSM 中,一条迁移被定义[14,16]为 $t=(s_i,x,s_j)$.其中,$f(s_i,x)=s_i,x\in I,s_i$ 为迁移 $t$ 的前状态,$s_j$ 为迁移 $t$ 的后状态,$x$ 是状态 $s_i$ 到状态 $s_j$ 的输入字符.为了描述方便,本文规定:若 $i=j$,则称 $t$ 为自迁移;若 $i\neq j$,则称 $t$ 为单迁移.

基于 FSM 模型的测试又被称为一致性测试(conformance testing)[1].该方法主要检测待测系统(system under

testing)是否与它的规格说明(FSM)相一致,即由 FSM 产生测试用例测试待测系统,测试者观察待测系统的运行是否完全符合它的 FSM 模型.由 FSM 产生测试用例的方法是遍历 FSM,获得 FSM 中一条或多条非空的连续迁移.本文采用迁移序列来描述连续迁移,并引入 Z 语言[15]中函数的值域操作符号 *ran* 和集合的长度操作符号#.符号 *ran* 和#同样适用于序列.例如,对于序列 $X=\langle t_1,t_2,t_3,t_1,t_2\rangle$,则 *ran* $X=\{t_1,t_2,t_3\}$和$\#X=5$.由于测试用例可用迁移序列来描述,为此,本文所指的测试用例集合均为迁移序列集合.




(a) A DFSM (b) A NDFSM

Fig.2 DFSM and NDFSM

图 2 DFSM 和 NDFSM

给定非空序列 $TS=\langle t_1,\ldots,t_n\rangle$,其中,$t_i$ 是 $M$ 中的迁移.若 $TS$ 中任意两个相邻的迁移 $t_i=(s_i,x_1,s_j)$和 $t_{i+1}=(s_p,x_2,s_q)$满足 $s_j=s_p$,则称 $TS$ 为 $M$ 的一条迁移序列.当$\#TS=1$,即 $TS=\langle t\rangle$和 $t=(s_1,x,s_2)$时:1) 若 $s_1=s_2$,则称 $TS$ 为 $M$ 的自迁移序列;2) 若 $s_1\neq s_2$,则称 $TS$ 为 $M$ 的单迁移序列.当$\#TS>1$ 时,称 $TS$ 为 $M$ 的多迁移序列.

与集合不同,迁移序列 $TS$ 中的迁移是有序的,如迁移序列$\langle t_i,t_{i+1}\rangle$与$\langle t_{i+1},t_i\rangle$并不相等.在图 1(a)中,$t_1=(1,c,1)$为自迁移,$t_2=(1,b,2)$和 $t_3=(1,a,3)$为单迁移,对应$\langle t_1\rangle$为自迁移序列,$\langle t_2\rangle$和$\langle t_3\rangle$为单迁移序列,$\langle t_1,t_2\rangle$和$\langle t_1,t_3\rangle$为多迁移序列.对于一个迁移序列 $TS$,*ran* $TS$ 表示 $TS$ 中所有迁移的集合.

**定义 2(相互独立)**. 设 $TS_1$ 和 $TS_2$ 为 $M$ 的两个迁移序列,若 *ran* $TS_1\cap$*ran* $TS_2=\varnothing$,则称 $TS_1$ 和 $TS_2$ 相互独立.

定义 2 表明,迁移序列 $TS_1$ 和 $TS_2$ 不包含相同迁移.

**定义 3(迁移覆盖)**. 假定 $M$ 的迁移集合为 $T=\{t_i|1\leqslant i\leqslant n\}$,集合 $A=\{TS_i|i\in\mathbb{N}\}$为 $M$ 中所有迁移序列的集合.若存在集合 $A_M\subseteq A$,$A_M\neq\varnothing$,使得 $\bigcup_{TS_l\in A_M} ran\,TS_l=T$, 则称 $A_M$ 为 $M$ 的迁移覆盖.

定义 3 借鉴了集合的覆盖定义,与集合的覆盖不同,迁移覆盖是由 $M$ 的迁移序列构成的集合.若由 $M$ 导出的测试用例集合是 $M$ 的一个迁移覆盖,则该测试用例集合是完备的;否则,必定存在 $M$ 中的某条迁移未被测试用例集合覆盖,导致测试用例集合的测试效率降低.

**定义 4(独立迁移覆盖)**. 给定 $M$ 的一个迁移覆盖 $A_{id}$,若$\forall TS_i,TS_j\in A_{id}$,都有 *ran* $TS_i\cap$*ran* $TS_j=\varnothing$,且$\forall TS\in A_{id}$ 都满足$\#(ran\ TS)=\#TS$,即集合 *ran* $TS$ 的大小等于序列 $TS$ 的长度,则称 $A_{id}$ 为 $M$ 的独立迁移覆盖.

由定义 4 可知,如果一个测试用例集合满足独立迁移覆盖,则测试用例集合中任意两个迁移序列都没有相同迁移,且任意一条迁移序列本身没有冗余迁移[14].

**定义 5(单迁移覆盖)**. 给定 $M$ 的一个迁移覆盖 $A_{sg}$,若$\forall TS_i\in A_{sg}$ 都满足$\#TS_i=1$,则称集合 $A_{sg}$ 为 $M$ 的单迁移覆盖.

因为$\#TS_i=1$,则 $TS_i$ 为自迁移序列或单迁移序列,于是有 $TS_i=\langle t_i\rangle$.其中,$t_i$ 为 $M$ 中的自迁移或单迁移.因此,$A_{sg}$ 是由 $M$ 中的所有单迁移序列或自迁移序列构成的.

**定理 1($A_{sg}$ 的独立性)**. 单迁移覆盖 $A_{sg}$ 为 $M$ 的独立迁移覆盖.

定理 1 显然成立.如果一个测试用例集合满足独立迁移覆盖,则该集合是完备的且无冗余迁移.为了得到 $M$ 中迁移集合 $T$ 的一个有效划分(即最小测试成本迁移覆盖),必须对 $A_{sg}$ 中的迁移序列进行连接,以达到减少集合中元素个数,同时不增加冗余迁移的目的.

**定义 6(连接)**. 若存在迁移序列 $TS_1=\langle t_1,t_2,\ldots,t_n\rangle$和 $TS_2=\langle t_l,t_{l+1},\ldots,t_{l+p}\rangle$,使得 $TS_1{}^\frown TS_2=\langle t_1,t_2,\ldots,t_n,t_l,t_{l+1},\ldots,t_{l+p}\rangle$,则称$^\frown$为迁移序列 $TS_1$ 对 $TS_2$ 的连接.

**定义 7(有效连接)**. 给定 $TS_1=\langle t_1,t_2,\ldots,t_n\rangle$和 $TS_2=\langle t_l,t_{l+1},\ldots,t_{l+p}\rangle$为 $M$ 的两条迁移序列,若 $TS_1^\frown TS_2$ 为 $M$ 的一条迁移序列,则 $TS_1$ 对 $TS_2$ 的连接是有效的.

**定理 2($A_{sg}$ 的连续性)**. 给定 $M$ 的单迁移覆盖 $A_{sg}$,若$\exists TS_1,TS_2\in A_{sg},TS_1=\langle t_1\rangle,TS_2=\langle t_2\rangle$,其中,迁移 $t_1=(s_i,x,s_j)$和 $t_2=(s_p,y,s_q)$满足 $s_j=s_p$,则 $M$ 中一定存在一个迁移序列 $TS$,使得 $TS=TS_1^\frown TS_2=\langle t_1,t_2\rangle$.

证明:将 $t_1$ 和 $t_2$ 的迁移函数 $f(s_i,x)=s_j$ 和 $f(s_p,y)=s_q$ 改写成 $f_x(s_i)=s_j$ 和 $f_y(s_p)=s_q$,由于 $s_j=s_p$,则将 $f_x(s_i)=s_j$ 代入 $f_y(s_p)=s_q$ 中,得到 $s_k=f_y(s_p)=f_y(f_x(s_i))=f_{x\circ y}(s_i)$,表示状态 $s_i$ 在 $x\circ y$(输入条件 $x$ 和 $y$ 的复合)的作用下迁移到状态 $s_j$,得到 $M$ 的迁移序列$\langle t_1,t_2\rangle$.令 $TS=\langle t_1,t_2\rangle$,则有 $TS=TS_1^\frown TS_2=\langle t_1,t_2\rangle$为 $M$ 中的一个迁移序列.证毕. □

定理 2 给出了单迁移覆盖 $A_{sg}$ 中的单迁移序列之间进行有效连接的条件.若对 $A_{sg}$ 中所有单迁移序列进行有效连接,得到的新集合仍为 $M$ 的一个覆盖.在不增加冗余迁移的情况下,新集合的大小要比 $A_{sg}$ 小很多.

**推论 1**. 若$\exists TS_1,TS_2\in A_{sg}$,使得 $TS=TS_1^\frown TS_2$ 为 $M$ 的一条迁移序列,则对于$\forall TS_3\in A_{sg},TS_1\neq TS_3,TS_2\neq TS_3$,都满足 $ran\ TS\cap ran\ TS_3=\varnothing$.

证明:由于$\exists TS_1,TS_2,TS_3\in A_{sg}$,则根据单迁移覆盖定义(定义 5),令 $TS_1=\langle t_1\rangle,TS_2=\langle t_2\rangle,TS_3=\langle t_3\rangle$.

由于 $TS=TS_1^\frown TS_2$ 为 $M$ 中的迁移序列,则根据有效连接定义(定义 7),$TS=\langle t_1,t_2\rangle$.

那么,$ran\ TS\cap ran\ TS_3=\{t_1,t_2\}\cap\{t_3\}=(\{t_1\}\cap\{t_3\})\cup(\{t_2\}\cap\{t_3\})=(ran\ TS_1\cap ran\ TS_3)\cup(ran\ TS_2\cap ran\ TS_3)$.

根据定理 1,有 $ran\ TS_1\cap ran\ TS_3=\varnothing$和 $ran\ TS_2\cap ran\ TS_3=\varnothing$,则 $ran\ TS\cap ran\ TS_3=\varnothing$.证毕. □

**推论 2**. 若$\exists TS_1,TS_2\in A_{id}$,使得 $TS=TS_1^\frown TS_2$ 为 $M$ 中的一个迁移序列,则对于$\forall TS_3\in A_{id},TS_1\neq TS_3,TS_2\neq TS_3$,都满足 $ran\ TS\cap ran\ TS_3=\varnothing$.

证明:由于$\exists TS_1,TS_2,TS_3\in A_{id}$,则令 $TS_1=\langle t_l,\ldots,t_m\rangle,TS_2=\langle t_o,\ldots,t_p\rangle,TS_3=\langle t_n,\ldots,t_q\rangle$.

由于 $TS=TS_1^\frown TS_2$ 为 $M$ 中的迁移序列,则 $TS=TS_1^\frown TS_2=\langle t_l,\ldots,t_m,t_o,\ldots,t_p\rangle$,则

$$\begin{aligned} ran\ TS\cap ran\ TS_3 &= \{t_l,...,t_m,t_o,...,t_p\}\cap\{t_n,...,t_q\}\\ &=(\{t_l,...,t_m\}\cap\{t_n,...,t_q\})\cup(\{t_o,...,t_p\}\cap\{t_n,...,t_q\})\\ &=(ran\ TS_1\cap ran\ TS_3)\cup(ran\ TS_2\cap ran\ TS_3). \end{aligned}$$

由于 $TS_1,TS_2,TS_3\in A_{id}$ 且 $TS_1\neq TS_2,TS_2\neq TS_3$,则根据独立迁移覆盖定义(定义 4),有 $ran\ TS_1\cap ran\ TS_3=\varnothing$和 $ran\ TS_2\cap ran\ TS_3=\varnothing$,则 $ran\ TS\cap ran\ TS_3=(ran\ TS_1\cap ran\ TS_3)\cup(ran\ TS_2\cap ran\ TS_3)=\varnothing\cup\varnothing=\varnothing$.证毕. □

根据定理 2 及推论 1 和推论 2,若 $A_{id}$ 中的迁移序列连接后仍然是 $M$ 的迁移序列,则可以用新生成的迁移序列替代被连接的迁移序列,得到新集合中迁移序列的数目比 $A_{id}$ 中迁移序列的数目要少很多,且迁移序列之间仍然相互独立.新集合中迁移序列的总长度为 $M$ 的迁移总数.

## 3 必要条件和优化迁移覆盖算法

**定义 8(最小测试成本迁移覆盖)**. 对于 $M$ 的一个迁移覆盖 $A_p$,满足下面两个条件:

(1) $A_p$ 是 $M$ 的一个独立迁移覆盖;

(2) 对于 $M$ 的任意一个独立迁移覆盖 $A_{id}$,都有$\#A_p\leqslant\#A_{id}$,

则称 $A_p$ 为 $M$ 的最小测试成本迁移覆盖.

最小测试成本迁移覆盖$A_p$是由集合的覆盖推导而来,并未考虑$A_p$中的测试序列的有效性[14],即测试序列能否由初态到达.本文将在第 4.4 节讨论这一问题.

由 $M$ 的独立迁移覆盖定义(定义 4)可知,$\forall TS_i,TS_j\in A_p,i\neq j\bullet(ran\ TS_i\cap ran\ TS_j=\varnothing\wedge(\#(ran\ TS_i)=\#TS_i))$;由 $M$ 的迁移覆盖定义(定义 3)可知,$A_p\neq\varnothing\wedge\bigcup_{TS_i\in A_p} ran\ TS_i=T$. 因此,由集合的划分定义可知,$A_p$ 也是 $M$ 中迁移集合 $T$ 的一个有效划分.由于 $A_p$ 覆盖了迁移集合 $T$(完备的),$A_p$ 中迁移序列之间相互独立(无冗余的)且 $A_p$ 中迁移序列的数目最少,则 $A_p$ 中迁移序列总长度为 $M$ 中的迁移总数.因此,$A_p$ 是 $M$ 的所有迁移覆盖中迁移序列总长度最短的集合之一.若测试用例集合满足 $A_p$,则该集合的测试成本最小.

说明 1:$A_p$ 中迁移序列的数目与 $M$ 中自迁移序列无关.如在图 3 中,$M$ 的最小测试成本覆盖为 $A_p=\{TS_1,TS_2\}$,其中,$TS_1=\langle t_1,t_2,t_4,t_5\rangle$和 $TS_2=\langle t_3\rangle$或者 $TS_1=\langle t_2,t_4,t_5\rangle$和 $TS_2=\langle t_1,t_3\rangle$.$M$ 中的自迁移序列$\langle t_1\rangle$并不改变 $A_p$ 的大小,仅改变

$A_p$中迁移序列的构成.为了构造集合$A_p$,必须研究集合$A_p$存在的必要条件和充分条件.

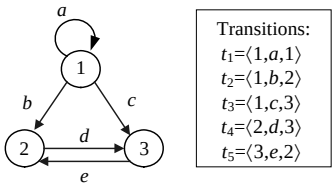



Fig.3 A DFSM and its transitions

图 3 一个 DFSM 及其迁移

### 3.1 必要条件

**定理 3(必要条件)**. 给定$\mathbb{P}(TS)$为$M$中独立迁移覆盖的幂集,集合$A_p\in\mathbb{P}(TS)$.如果$A_p$为$M$的最小测试成本覆盖,则对于$\forall TS_1\in A_p$,都不存在$TS_2\in A_p$使得$TS=TS_1^\frown TS_2$,或$TS=TS_2^\frown TS_1$为$M$中的一条迁移序列.

证明(反证法):

假设集合$A_p\in\mathbb{P}(TS)$为$M$的最小测试成本覆盖,且$\exists TS_1,TS_2\in A_p$,使得$TS=TS_1^\frown TS_2$,或$TS=TS_2^\frown TS_1$为$M$中的一个迁移序列.

令集合$A'_p=(A_p-\{TS_1,TS_2\})\cup TS$.

因为$A_p\in\mathbb{P}(TS)$,所以$A_p$为$M$的一个独立迁移覆盖.

根据推论 2,对于$\forall TS_3\in A_p, TS_1\neq TS_3, TS_2\neq TS_3$,都有$ran\ TS\cap ran\ TS_3=\varnothing$.

根据独立迁移覆盖定义(定义 4),$A'_p$为$M$的一个独立迁移覆盖.

由于$A'_p=(A_p-\{TS_1,TS_2\})\cup TS, \#\{TS_1,TS_2\}=2, \#TS=1$,则$\#A_p=\#A'_p+1$.

由最小测试成本迁移覆盖的定义(定义 8)可知,集合$A_p$不是$M$的最小测试成本迁移覆盖,假设错误,则定理 3 成立.证毕. □

由定理 1 可知,若测试用例集合满足$M$的单迁移覆盖$A_{sg}$,则测试用例集合中的迁移序列相互独立.根据定理 2 和定理 3,将$A_{sg}$中单迁移序列进行有效连接操作,可以减少集合中迁移序列的数目.

**定义 9(优化迁移覆盖)**. 给定$A_{sg}$为$M$的单迁移覆盖,对$A_{sg}$中所有满足有效连接的单迁移序列进行连接,直到集合中所有迁移序列都不能进行有效连接为止,最终得到的集合$A_{ad}$被称为$M$的优化迁移覆盖.

说明 2:$M$的优化迁移序列覆盖$A_{ad}$并不唯一.例如,对于图 3 中单迁移序列$\langle t_1\rangle,\langle t_2\rangle,\langle t_3\rangle,\langle t_4\rangle$和$\langle t_5\rangle$,合并所有有效连接操作的单迁移序列,生成优化迁移序列覆盖$A_{ad}=\{\langle t_1,t_2,t_4\rangle,\langle t_3,t_5\rangle\}$或$A_{ad}=\{\langle t_2,t_4\rangle,\langle t_1,t_3,t_5\rangle\}$.

### 3.2 优化迁移覆盖算法

根据优化迁移覆盖(定义 9),本节设计了优化迁移覆盖算法.算法的主要思想是:

1) 将$M$中所有自迁移序列加入到输出集合$SS$中,并从迁移集合$T$中删除所有自迁移;
2) 从迁移集合$T$中取一个单迁移$t$,令序列$TS=\langle t\rangle$及$T=T\backslash t$,若$t_i\in T$使得$TS^\frown\langle t_i\rangle$为有效连接操作,则$TS=TS^\frown\langle t_i\rangle$及$T=T\backslash t_i$,直到$T$中不存在这样的$t_i$;
3) 若$TS$可与$SS$中的迁移序列进行有效连接操作,则对它们进行有效连接操作,否则,将单迁移序列$TS$加入到$SS$中;
4) 若$T$不空,则转到步骤 2).

**算法 1**. 优化迁移覆盖算法.

输入:存储 DFSM 的邻接矩阵,迁移集合$T=\{t_1,\ldots,t_n\}$和迁移约束集$TC$;

输出:满足优化迁移序列覆盖的集合$SS$.

1. 初始化集合$SS=\{\}$,迁移序列$TS=null$.

2. 对每个自迁移 $t\in T$,令 $SS=SS\cup\langle t\rangle$,$T=T\backslash t$.
3. while ($T$!=*null*)
4. Get the first transition $t$ in $T$;
5. $TS=\langle t\rangle$;
6. $T=T\backslash t$;
7. for each $t_i\in T$
8. if ($TS^\frown\langle t_i\rangle$ in $M$) then
9. $TS=TS^\frown\langle t_i\rangle$; $T=T\backslash t_i$;
10. end if
11. end for
12. for each $TS_i\in SS$
13. if ($TS_i^\frown TS$ or $TS^\frown TS_i$ in $M$) then
14. $TS=TS_i^\frown TS$ or $TS=TS^\frown TS_i$; $SS=SS\backslash TS_i$;
15. end if
16. end for
17. $SS=SS\cup TS$; $TS=null$;
18. end while
19. Return $SS$.

假定迁移集合 $T$ 有 $n$ 条迁移、$k$ 条自迁移,则算法 1 的时间复杂度为 $O(kn^2-k^2n)$;若 $k$ 是一个常数,且 $n$ 远大于 $k$,则算法 1 的时间复杂度为 $O(n^2)$.

### 3.3 例 子

本节采用了图 4 中的例子演示了优化迁移覆盖算法,其中,迁移集合为 $T=\{t_i|0\leqslant i\leqslant 8\}$.为了直观描述,在图 4 中的 FSM 上直接标注了迁移.优化迁移覆盖算法的具体演示步骤见表 1.

**Table 1** Process of building the set *SS*

**表 1** 集合 *SS* 的构造过程

| Operations | *TS* | *T* | *SS* |
|---|---|---|---|
| Initial | *Null* | $\{t_i\|0\leqslant i\leqslant 8\}$ | {} |
| Self transitions | *Null* | $\{t_1,t_3,t_5,t_6,t_7,t_8\}$ | $\{\langle t_0\rangle,\langle t_2\rangle,\langle t_4\rangle\}$ |
| Get $t_1$ from $T$ | $\langle t_1\rangle$ | $\{t_3,t_5,t_6,t_7,t_8\}$ | $\{\langle t_0\rangle,\langle t_2\rangle,\langle t_4\rangle\}$ |
| If $TS^\frown\langle t_i\rangle$ in $M$ | $\langle t_1,t_3\rangle$ | $\{t_5,t_6,t_7,t_8\}$ | $\{\langle t_0\rangle,\langle t_2\rangle,\langle t_4\rangle\}$ |
| For each $TS_i\in SS$, $TS=TS_i^\frown TS$ or $TS=TS^\frown TS_i$, $TS$ in $M$ | $\langle t_0,t_1,t_3,t_4\rangle$ | $\{t_5,t_6,t_7,t_8\}$ | $\{\langle t_2\rangle\}$ |
| $SS=SS\cup TS$, $TS=null$ | *Null* | $\{t_5,t_6,t_7,t_8\}$ | $\{\langle t_2\rangle,\langle t_0,t_1,t_3,t_4\rangle\}$ |
| Get $t_1$ from $T$ | $\langle t_5\rangle$ | $\{t_6,t_7,t_8\}$ | $\{\langle t_2\rangle,\langle t_0,t_1,t_3,t_4\rangle\}$ |
| If $TS^\frown\langle t_i\rangle$ in $M$ | $\langle t_5,t_6\rangle$ | $\{t_7,t_8\}$ | $\{\langle t_2\rangle,\langle t_0,t_1,t_3,t_4\rangle\}$ |
| For each $TS_i\in SS$, $TS=TS_i^\frown TS$ or $TS=TS^\frown TS_i$, $TS$ in $M$ | $\langle t_2,t_5,t_6\rangle$ | $\{t_7,t_8\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle\}$ |
| $SS=SS\cup TS$, $TS=null$ | *Null* | $\{t_7,t_8\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_2,t_5,t_6\rangle\}$ |
| Get $t_1$ from $T$ | $\langle t_7\rangle$ | $\{t_8\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_2,t_5,t_6\rangle\}$ |
| If $TS^\frown\langle t_i\rangle$ in $M$ | $\langle t_7\rangle$ | $\{t_8\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_2,t_5,t_6\rangle\}$ |
| For each $TS_i\in SS$, $TS=TS_i^\frown TS$ or $TS=TS^\frown TS_i$, $TS$ in $M$ | $\langle t_7,t_2,t_5,t_6\rangle$ | $\{t_8\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle\}$ |
| $SS=SS\cup TS$, $TS=null$ | *Null* | $\{t_8\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle\}$ |
| Get $t_1$ from $T$ | $\langle t_8\rangle$ | *Null* | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle\}$ |
| If $TS^\frown\langle t_i\rangle$ in $M$ | $\langle t_8\rangle$ | *Null* | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle\}$ |
| For each $TS_i\in SS$, $TS=TS_i^\frown TS$ or $TS=TS^\frown TS_i$, $TS$ in $M$ | $\langle t_8\rangle$ | *Null* | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle\}$ |
| $SS=SS\cup TS$, $TS=null$ | *Null* | *Null* | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle,\langle t_8\rangle\}$ |
| Return *SS* | | | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle,\langle t_8\rangle\}$ |

由表 1 可知,优化迁移覆盖算法产生集合 $SS=\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle,\langle t_8\rangle\}$.然而,集合 $SS$ 并不满足 $M$ 的最小测试成本迁移覆盖,如另一组迁移序列集合 $SS_1=\{\langle t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle,\langle t_8\rangle\}$,使得$\#SS_1<\#SS$.根据最小测试成本迁移覆盖(定义 8),集合 $SS$ 并不是 $M$ 的最小测试成本迁移覆盖.因此,定理 3 中的必要条件并不能保证得到 $M$ 的最小测试成本迁移覆盖,需要研究 $M$ 的最小测试成本迁移覆盖的充分条件.

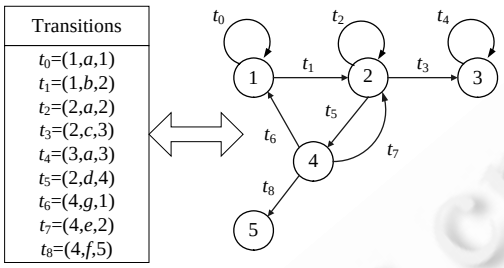



Fig.4 A case for algorithm of optimal transition coverage

图 4 优化迁移序列覆盖算法的示例

## 4 充分条件和最小测试成本迁移覆盖算法

### 4.1 充分条件

**定义 10(相交序列)**. 在 $M$ 中,对于任意两条迁移序列 $TS_1$ 和 $TS_2$,若 $TS_1$ 和 $TS_2$ 中的迁移包含了某些共同状态,则称迁移序列 $TS_1$ 和 $TS_2$ 为相交序列,这些共同状态为 $TS_1$ 和 $TS_2$ 的相交点;否则,称 $TS_1$ 和 $TS_2$ 为不相交序列.

为了得到 $M$ 的最小测试成本迁移覆盖,需要分析迁移序列之间的关系.$M$ 中的任意两个迁移序列 $TS_1$ 和 $TS_2$ 可能是相交序列或不相交序列.若 $TS_1$ 和 $TS_2$ 不相交,则 $M$ 中一定不存在一个迁移序列 $TS$ 能够一次遍历完 $TS_1$ 和 $TS_2$,且同时满足$\#TS=\#TS_1+\#TS_2$.因此,只有在 $TS_1$ 和 $TS_2$ 是相交序列的情况下,$M$ 中才可能存在一个迁移序列 $TS$,使得 $TS$ 一次遍历了 $TS_1$ 和 $TS_2$ 中所有迁移且满足$\#TS=\#TS_1+\#TS_2$.

**定义 11(序列合并)**. 在 $M$ 中,对于任意两个迁移序列 $TS_1$ 和 $TS_2$,若 $M$ 中存在一个迁移序列 $TS$,$TS$ 由 $TS_1$ 和 $TS_2$ 中所有迁移构成,并且 $TS_1$ 和 $TS_2$ 中的所有迁移在 $TS$ 中出现且仅出现 1 次,则称迁移序列 $TS$ 为迁移序列 $TS_1$ 和 $TS_2$ 的合并.

序列合并的实质是,一次遍历完两条迁移序列中的所有迁移,且不重复遍历任何迁移.迁移序列的连接操作是序列合并的一种特殊情况,还存在其他情况.如图 4 中$\langle t_1,t_3\rangle$和$\langle t_2\rangle$为 $M$ 中的两个迁移序列,而$\langle t_1,t_3\rangle^\frown\langle t_2\rangle$和$\langle t_2\rangle^\frown\langle t_1,t_3\rangle$都不是 $M$ 的一个迁移序列,但它们可以合并成 $M$ 中的一个迁移序列$\langle t_1,t_2,t_3\rangle$.

**定义 12(回路)**. 在 $M$ 中,$A_{id}$ 为 $M$ 的独立迁移序列覆盖,若$\exists TS\in A_{id}$,$TS=\langle t_1,\ldots,t_k\rangle$,$t_1=(s_1,i_1,s_l)$和 $t_k=(s_i,i_k,s_j)$,满足条件 $s_1=s_j$,则称迁移序列 $TS$ 为回路.

**定理 4(不可合并)**. 在 $M$ 中,设 $A_{id}$ 为 $M$ 的独立迁移序列覆盖,若$\exists TS_1,TS_2\in A_{id}$,使得下列条件之一:

(1) $TS_1$ 和 $TS_2$ 不相交;

(2) $TS_1$ 和 $TS_2$ 相交,则 $TS=TS_1^\frown TS_2$ 和 $TS=TS_2^\frown TS_1$ 均不是 $M$ 的迁移序列,并且 $TS_1$ 和 $TS_2$ 都不是回路,则迁移序列 $TS_1$ 和 $TS_2$ 不能被合并成 $M$ 中的一条迁移 $TS$.

证明:

(1) 如果 $TS_1$ 和 $TS_2$ 不相交,显然,$TS_1$ 和 $TS_2$ 不能合并成 $M$ 中的一个迁移;

(2) 如果 $TS_1$ 和 $TS_2$ 相交,由于 $TS_1,TS_2\in A_{id}$,则 $ran\ TS_1\cap ran\ TS_2=\varnothing$,即迁移序列 $TS_1$ 和 $TS_2$ 中没有共同的迁移.因此,$TS_1$ 和 $TS_2$ 只能相交于一个或多个点(状态).相交点在两条序列中的位置有如下 3 种情况:

① 交点位于迁移序列 $TS_1$ 或 $TS_2$ 的初始位置;

② 交点位于迁移序列 $TS_1$ 或 $TS_2$ 的结束位置;

③ 交点位于迁移序列 $TS_1$ 和 $TS_2$ 的中间位置.

图 5 显示了相交点位于迁移序列 $TS_1$ 和 $TS_2$ 不同位置的抽象模型.

图 5(a)和图 5(c)的抽象模型满足 $TS=TS_1^\frown TS_2$ 或 $TS=TS_2^\frown TS_1$ 是 $M$ 的一个迁移序列,因此可以合并.而图 5(b)、图 5(d)和图 5(e)需要考虑迁移序列 $TS_1$ 和 $TS_2$ 是否为回路.

若 $TS_1$ 和 $TS_2$ 都不是回路,且不满足 $TS=TS_1^\frown TS_2$ 或 $TS=TS_2^\frown TS_1$ 是 $M$ 的一个迁移序列,则由几何学的知识可知,在不增加冗余迁移的情况下,迁移序列 $TS_1$ 和 $TS_2$ 无法一次被遍历完成.

若 $TS_1$ 或 $TS_2$ 包含回路:

在图 6(a)和图 6(b)中,抽象模型满足 $TS=TS_1^\frown TS_2$ 或 $TS=TS_2^\frown TS_1$ 是 $M$ 的迁移序列,因此,$TS_1$ 或 $TS_2$ 能够合并成 $M$ 中的一个迁移;在图 6(c)中,不妨假设迁移序列 $TS_1=\langle t_1,t_2,t_3\rangle$和 $TS_2=\langle t_0,t_4\rangle$,其中,$t_1=(s_1,i_1,s_2)$,$t_2=(s_2,i_2,s_3)$,$t_3=(s_3,i_3,\ s_1)$,$t_0=(s_0,i_0,s_2)$,$t_4=(s_2,i_4,s_4)$,$s_2$ 是 $TS_1$ 和 $TS_2$ 的一个相交点,则 $TS_1$ 包含相交点 $s_2$ 的回路.迁移序列 $TS_1$ 和 $TS_2$ 可以合并成迁移序列 $TS=\langle t_0,t_2,t_3,t_1,t_4\rangle$.

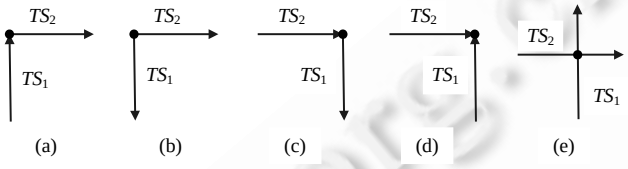



Fig.5 Abstract model for the position of point of intersection in $TS_1$ and $TS_2$

图 5 $TS_1$ 和 $TS_2$ 相交点位置的抽象模型

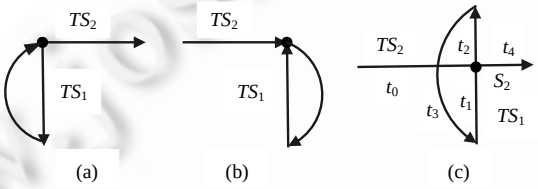



Fig.6 Abstract model of the existing loop in $TS_1$ or $TS_2$

图 6 $TS_1$ 或 $TS_2$ 存在回路的抽象模型

综上所述,定理 4 成立. □

**定理 5(充分条件)**. 给定 $M$ 中独立迁移覆盖 $A_{id}$ 的幂集为$\mathbb{P}(TS)$,若$\exists A_{op}\in\mathbb{P}(TS)$,使得$\forall TS_1,TS_2\in A_{op}$,迁移序列 $TS_1$ 和 $TS_2$ 都不可以合并,则 $A_{op}$ 为 $M$ 的最小测试成本迁移覆盖.

证明(归纳法):

(1) 假设集合 $A_{op}$ 中有且仅有一个迁移序列 $TS$,则集合 $A_{op}$ 中迁移序列的数目最少,因此,

$$\forall A_{id}\in\mathbb{P}(TS)\bullet(\#A_{id}\leqslant\#A_{op}).$$

(2) 假设集合 $A_{op}$ 中仅有两个迁移序列 $TS_1$ 和 $TS_2$,且它们不可以合并.由定理 4 可知,迁移序列 $TS_1$ 和 $TS_2$ 不能被合并成 $M$ 中的一条迁移 $TS$,因此,$\forall A_{id}\in\mathbb{P}(TS)\bullet(\#A_{id}\leqslant\#A_{op})$.

(3) 假设集合 $A_{op}=\{TS_1,\ldots,TS_k\}$为 $M$ 的最小测试成本迁移覆盖,再假设当集合 $A'_{op}=\{TS_1,\ldots,TS_k,TS_{k+1}\}$时,对于$\forall TS_i,TS_j\in A'_{op}$, $TS_i$ 和 $TS_j$ 不可以合并.由于 $A_{op}=\{TS_1,\ldots,TS_k\}$是 $M$ 的最小测试成本迁移覆盖,则集合 $A_{op}$ 中迁移序列的数目为 $k$.若集合 $A'_{op}=\{TS_1,\ldots,TS_k,TS_{k+1}\}$不是 $M$ 的最小测试成本迁移覆盖.由于 $A'_{op}=A_{op}\cup TS_{k+1}$且 $A_{op}$ 为 $M$ 的最小测试成本迁移覆盖,则必有迁移序列 $TS_{k+1}$ 可以与 $TS_1,\ldots,TS_k$ 中的某个迁移序列合并.这与假设对于$\forall TS_i,TS_j\in A'_{op}$, $TS_i$ 和 $TS_j$ 不可以合并矛盾.因此, $A'_{op}=\{TS_1,\ldots,TS_k,TS_{k+1}\}$ 为 $M$ 的最小测试成本迁移覆盖.

综合情况(1)~情况(3),可知定理 5 成立. □

根据定理 5,在优化迁移覆盖算法的基础上,本文设计了 $M$ 的最小测试成本迁移覆盖的实现算法.

### 4.2 最小测试成本迁移覆盖算法

由定理 5 可知,只要独立迁移覆盖 $A_{id}$ 中的任意两个迁移序列都不能合并,则这个独立迁移覆盖 $A_{id}$ 就是 $M$ 的最小测试成本迁移覆盖.因此,最小测试成本迁移覆盖算法的主要思想是:针对优化迁移覆盖算法产生的输出集合 $SS$,合并该集合中所有可合并的迁移序列,直到任意两个迁移序列都不能合并为止.为此,本文设计了迁移序列合并的准则:

**准则 1**. 若 $M$ 中迁移序列 $TS_1,TS_2\in A_{id},TS_1$ 和 $TS_2$ 相交,且满足 $TS=TS_1\frown TS_2$ 或 $TS=TS_2\frown TS_1$ 为 $M$ 的迁移序列,则用迁移序列 $TS$ 替代迁移序列 $TS_1$ 和 $TS_2$.

**准则 2**. 假设 $M$ 中迁移序列 $TS_1,TS_2\in A_{id},TS_1=\langle t_1,t_2,\ldots,t_{i-1},t_i,\ldots,t_k\rangle$ 与 $TS_2=\langle t_m,t_{m+1},\ldots,t_{j-1},t_j,\ldots,t_n\rangle$ 的相交点为 $s_i$,其中,迁移 $t_{i-1}=(s_{i-1},i_{i-1},s_i)$ 和 $t_j=(s_i,i_j,s_j)$,且 $TS_2$ 是回路,则迁移序列 $TS_1$ 和 $TS_2$ 合并成迁移序列

$$TS=\langle t_1,t_2,\ldots,t_{i-1},t_j,\ldots,t_n,t_m,t_{m+1},\ldots,t_{j-1},t_i,\ldots,t_k\rangle.$$

**算法 2**. 最小测试成本迁移覆盖算法.

输入:优化迁移覆盖算法的输出集合 $SS$;

输出:满足 $M$ 的最小测试成本迁移覆盖的序列集合 $SS_1$.

1. 初始化状态集合 $SS_1=Null$;
2. while ($SS!=Null$)
3. 取 $SS$ 中第 1 个迁移序列 $TS_k$,$SS=SS\backslash TS_k$;
4. if ($SS_1=Null$) then $SS_1=SS_1\cup\{TS_k\}$;
5. else for each $TS_i\in SS_1$
6. if ($TS_k$ 与 $TS_i$ 不相交) then $SS_1=SS_1\cup TS_k$;
7. else if ($TS=TS_k\frown TS_i$ 或 $TS=TS_i\frown TS_k$ 为 $M$ 中的迁移序列) then //准则 1
8. $SS_1=SS_1\backslash TS_i$; $SS_1=SS_1\cup TS$;
9. else if ($TS_k$ 或 $TS_i$ 是回路) then //准则 2
10. 取相交点 $s_i$,根据 $s_i$ 将迁移序列 $TS_k$ 分为 $TS_{k-1}$ 和 $TS_{k-2}$,根据 $s_i$ 将迁移序列 $TS_i$ 分为 $TS_{i-1}$ 和 $TS_{i-2}$;
11. if ($TS_i$ 存在相交点的回路) then
12. $SS_1=SS_1\cup(TS_{k-1}\frown TS_{i-2}\frown TS_{i-1}\frown TS_{k-2})$;
13. else
14. $SS_1=SS_1\cup(TS_{i-1}\frown TS_{k-2}\frown TS_{k-1}\frown TS_{i-2})$;
15. end if
16. else $SS_1=SS_1\cup TS_k$;
17. end if
18. end if
19. end if
20. end for
21. end if
22. end while
23. return $SS_1$;

令 FSM 中有 $n$ 条迁移,集合 $SS$ 中有 $m$ 个元素,则算法 2 的时间复杂度要小于 $O(m^2n)$.

### 4.3 例 子

本节采用第 3.3 节中的例子演示最小测试成本迁移覆盖算法.在第 3.3 节中生成的满足优化迁移序列覆盖的状态序列集合为 $SS=\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle,\langle t_8\rangle\}$,$M$ 的最小测试成本迁移覆盖 $SS_1$ 的具体产生过程见表 2.

由表 2 可知,最终生成满足 $M$ 的最小测试成本迁移覆盖 $SS_1=\{\langle t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle,\langle t_8\rangle\}$,该集合中的迁移序列

不可合并,且不存在冗余迁移.

说明 3:由于自迁移序列$\langle t_0\rangle$,$\langle t_2\rangle$和$\langle t_4\rangle$会改变迁移序列的构成,因此,最小测试成本迁移覆盖算法生成的集合$SS_1$并不唯一.

**Table 2** Process of building the set $SS_1$

**表 2** 集合 $SS_1$ 的构造过程

| Operations | $TS_k$ | $SS$ | $SS_1$ |
|---|---|---|---|
| Initial | | $\{\langle t_0,t_1,t_3,t_4\rangle,\langle t_7,t_2,t_5,t_6\rangle,\langle t_8\rangle\}$ | *Null* |
| Get first $TS_k\in SS$ | $\langle t_0,t_1,t_3,t_4\rangle$ | $\{\langle t_7,t_2,t_5,t_6\rangle,\langle t_8\rangle\}$ | *Null* |
| For each $TS_i\in SS_1$ | $\langle t_0,t_1,t_3,t_4\rangle$ | $\{\langle t_7,t_2,t_5,t_6\rangle,\langle t_8\rangle\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle\}$ |
| Get first $TS_k\in SS$ | $\langle t_7,t_2,t_5,t_6\rangle$ | $\{\langle t_8\rangle\}$ | $\{\langle t_0,t_1,t_3,t_4\rangle\}$ |
| For each $TS_i\in SS_1$ | $\langle t_7,t_2,t_5,t_6\rangle$ | $\{\langle t_8\rangle\}$ | $\{\langle t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle\}$ |
| Get first $TS_k\in SS$ | $\langle t_8\rangle$ | *Null* | $\{\langle t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle\}$ |
| For each $TS_i\in SS_1$ | $\langle t_8\rangle$ | *Null* | $\{\langle t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle,\langle t_8\rangle\}$ |
| Return $SS_1$ | | | $\{\langle t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle,\langle t_8\rangle\}$ |

### 4.4 有效性讨论

若图 4 所示 FSM 的初始状态 $s_0$ 为节点 1,则 $SS_1=\{\langle t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle,\langle t_8\rangle\}$中的两条测试序列均不能由初始状态 $s_0$ 到达,因此,这两条测试序列是无效的测试片段(ineffective test segment),文献[14]给出了无效测试片段转换成有效测试序列的转换规则.因为 $t_7=(4,e,2)$,$t_8=(4,f,5)$,$s_0$ 为节点 1,因此仅需找到一条由节点 1 到节点 4 的最短路径(由测试序列表示),然后将这条最短路径与 $SS_1$ 中的测试序列进行连接,则无效测试片段就能转换成有效的测试序列.同时,生成的测试用例集合仍然满足最小测试成本迁移覆盖.获得图中两点间最短路径的方法有很多,本文采用 on-the-fly 的方式,广度优先遍历 FSM 获得两点间最短路径,具体过程如图 7 所示.

步骤 1. 剔除所有的自循环;

步骤 2. 广度优先遍历,当遍历到节点 4 时,遍历结束.那么,节点 1 到节点 4 的最短路径为$\langle t_1,t_5\rangle$.

因此,最终获得满足最小测试成本覆盖的集合为 $SS_2=\{\langle t_1,t_5,t_7,t_2,t_5,t_6,t_0,t_1,t_3,t_4\rangle,\langle t_1,t_5,t_8\rangle\}$.虽然 $SS_2$ 中包含少量的冗余,造成测试成本的增加,但根据文献[14]的结论,测试用例集合中的部分冗余是必要的.为了分析测试用例集合 $SS_2$ 的测试成本的增长情况,本文将在第 5 节进行实验评估.

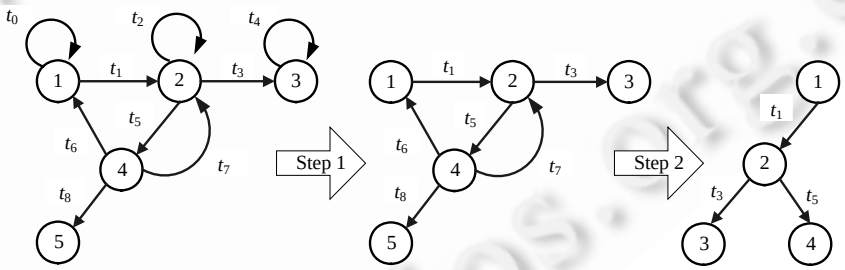



Fig.7 Shortest path between two nodes

图 7 两点间最短路径

## 5 实验对比

本节以在线邮箱系统(GeneRal gmAiL,简称 GRAL)为例,将本文的方法与 W 方法、U 方法和 R-Wp 方法进行了比较.首先,本文比较了这些方法的错误探测能力,随后对比了由这些方法产生的迁移序列集合的大小、迁移总数和冗余数目.为了客观和公正地进行实验对比,本文从最新的一些文献[12,16,17]中选用了 W 方法、U 方法和 R-Wp 方法的改进方法.为了比较错误探测能力,我们在 GRAL 系统中植入了两类错误:一类错误在页面的

JavaScript 代码中,如用户密码本应由 6~12 位的字符构成,但被误写为由 0~12 位的字符构成;第 2 类错误在 Java 代码中,如时间格式本应为“年-月-日”,但被误写为“年-月日”.所有这些错误都不影响系统的实际运行.

### 5.1 实验对象

图 8(a)为 GRAL 系统的页面功能模型.图 8(b)是该系统的 FSM 模型,其中,$P_1$ 表示 login(初始状态),$P_2$ 表示 Gmail Home,$P_3$ 表示 Create user,$P_4$ 表示 Psd help,$P_5$ 表示 Cr us with reminder,$P_6$ 表示 Gmail Introduction,$P_7$ 表示 Other user.图 8(c)为植入错误后的 GRAL 系统的 FSM 模型,其中:link 表示预期状态 $P_4$ 没有出现;success 表示 $P_5$ 中创建用户成功后,错误地进入了 $P_2$;Username check 表示 $P_5$ 中用户名的检测无效;error 表示 $P_5$ 中“用户名+密码+验证码+基本资料(可选项)”的输入未被检测;$\underline{P_6}$ 表示 $P_6$ 中显示的时间格式不对;$\underline{P_2}$ 表示经过 $P_5$ 的 success 操作后,$P_2$ 获得 $P_6$ 发送的欢迎新用户邮件的错误.

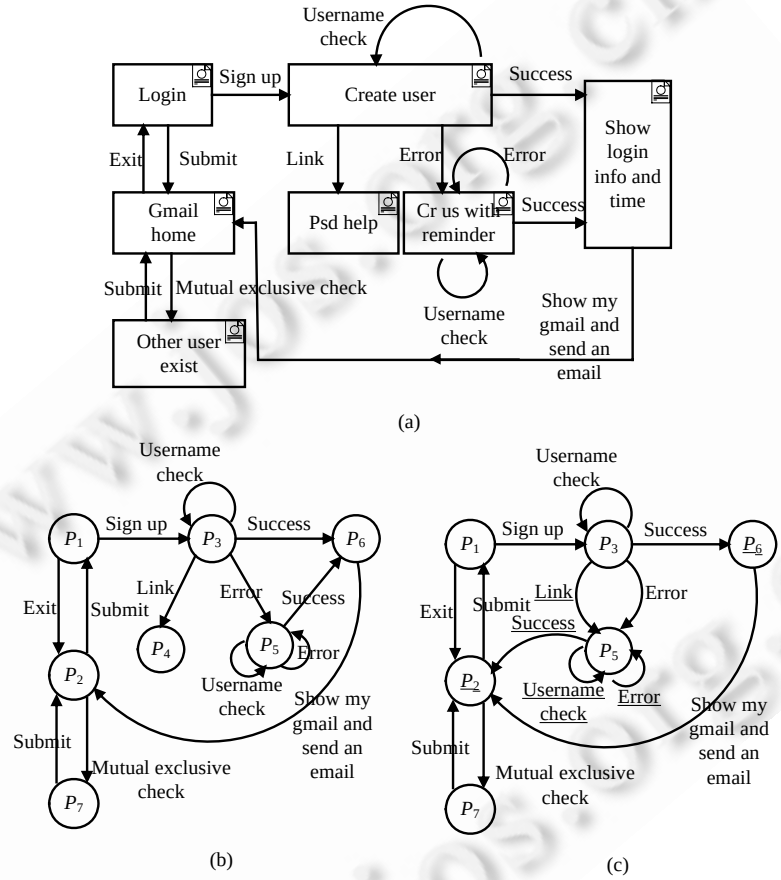



Fig.8 Three models of GRAL

图 8 GRAL 系统的 3 个模型

### 5.2 实验方法

Chow[3]提出的 W 方法需要构造 $P$ 集和 $Z$ 集,其中,$P$ 集由一棵测试树(testing tree)产生,$Z$ 集是特征集 $W$ 和 FSM 的输入条件集 $X$ 的并.由于该方法需要构造 $Z$ 集,且测试树还包含了空条件,导致最终得到的测试输入序列数目过多.为此,许多研究人员都提出了改进的方法.我们提出了一种简化的思想[17].该方法生成一棵广度优先生成树,树的节点为 FSM 中的状态,根节点为初始状态,树的边为 FSM 中的非空输入.测试序列为根到所有叶子节点的序列,序列的组成为“根.边.节点.边…叶子”.我们的方法既不用构造 $Z$ 集,又能有效地减少测试输入序列

的数目,本节采用这种改进的 W 方法并进行了实验比较.

Ural 等人[18]提出的 U 方法是目前针对协议测试的一种主要方法,该方法首先为状态机的每一个状态得到一个识别序列 UIO,然后构造测试输入序列.这种方法构造的测试序列存在过多的冗余[16],因此,Duan 和 Chen[16]采用了旅行者问题的解决方法,利用序列中的冗余部分合并测试序列,以达到减少测试序列的数目及冗余的目的.本节采用这种改进的 U 方法进行实验比较.为了方便比较,本文假定 GRAL 模型中的迁移序列均是唯一输入/输出序列.

张涌等人[12]改进了 Wp 方法[9],提出了 R-Wp 方法.该方法通过排除 Wp 方法中错误的状态迁移,而减少测试输入序列的数目.R-Wp 方法包含如下 3 个步骤:

(1) 估计实现中的有限状态机中状态数目的上界 $m$,其中,$m$ 要大于或等于规约说明中的有限状态机中的状态数目 $n$;

(2) 检验在规约说明中的状态在实现中可被识别并验证其正确性,同时从初始状态到这些状态所经历的状态迁移也被验证,该部分的测试输入序列可如下构造:$T_1=Q.X[m-n].W$;

(3) 验证在上面未被验证的状态迁移,测试输入序列可构造如下:

$$T_2=(P-Q)\otimes W=\bigcup_{p_1\in(P-Q)}\{p_1\}.W_j,$$

其中,$Q$ 是 $S$ 的状态覆盖集,$P$ 是 $S$ 的状态迁移覆盖集,$W$ 是有限状态机的特征集,具体的构造方法参见文献[3,12].

为了比较的客观性,我们实现的 R-Wp 方法并不包括用于验证额外状态而产生的测试序列.

### 5.3 错误探测能力的讨论

为了比较本文的方法与 W 方法、U 方法和 R-Wp 方法的错误探测能力,需要针对由这些方法得到的迁移序列进行实例化(设计测试用例).本文采用手工方式为迁移序列设计了相应的测试用例,限于篇幅,迁移序列的实例化过程被略去,具体方法可参考文献[19].在测试植入错误的 GRAL 系统时,发现满足上述测试覆盖准则的测试用例集合均能发现 GRAL 系统的出错状态 $P_4$,$P_5$,$P_6$ 和 $P_2$.

### 5.4 实验结果及评估

表 3 显示了本文的方法与 W 方法、U 方法和 R-Wp 方法在 GRAL 中的实验对比结果,其中,$A_{ad}$ 表示优化迁移覆盖,$A_{op}$ 表示 $M$ 的最小测试成本迁移覆盖,$A_{ef}$ 为满足 $A_{op}$ 且初态可达的有效测试用例集合.实验对比项目包括测试用例集合的大小、迁移总数和冗余迁移的数目.依据表 3,一些重要的结论可以得出:

1) 由 W 方法、U 方法、R-Wp 方法和本文的方法获得的测试用例集合都能发现 GRAL 系统的出错状态 $P_4$,$P_5$,$P_6$ 和 $P_2$,因此,这些测试覆盖准则都能探测到迁移中的错误;

2) 在 $A_{ad}$ 和 $A_{op}$ 中,迁移总数最少且不存在冗余迁移,这表明我们的方法能够有效地减少测试序列中的冗余;

3) 对于 U 方法,$A_{op}$ 和 $A_{ef}$ 得到的测试用例集合中元素最小,这说明,对于 U 方法,$A_{op}$ 和 $A_{ef}$ 都能大幅减少测试用例的数目,降低测试成本;

4) 对比 $A_{op}$ 我们发现,$A_{ef}$ 仅仅增加了两条冗余的迁移,使得 $A_{ef}$ 中所有迁移序列均是有效的(初始状态为 $P_1$);对比 W 方法、U 方法和 R-Wp 方法,$A_{ef}$ 的集合大小、迁移总数和冗余数目都是最小的,因此,$A_{ef}$ 的测试成本最低.

**Table 3** Experimental comparison of several test coverage criterias

**表 3** 多个测试覆盖准则的实验比较

| Items | W-Method | U-Method | R-Wp method | $A_{ad}$ | $A_{op}$ | $A_{ef}$ |
|---|---|---|---|---|---|---|
| Sizes of set | 8 | 3 | 6 | 6 | 3 | 3 |
| Number of all transitions | 21 | 16 | 20 | 13 | 13 | 15 |
| Redundancy | 8 | 3 | 7 | 0 | 0 | 2 |

### 5.5 实验有效性讨论

1) 虽然实验选取的在线邮箱系统较小,但该系统包括自迁移、单迁移和迁移循环,因此具有一定代表性;
2) 在选取 W 方法、U 方法和 R-Wp 方法时,本文采用了最近一些文献中提到的改进方法,因此实验结果是公正、客观及可信的;
3) 由实验结果可知,在覆盖 GRAL 系统中的所有迁移时,本文方法能够获得测试成本最小的迁移序列集合.

## 6 相关工作比较

本节主要从基于 FSM 的测试覆盖准则和状态组合爆炸问题两个方面,对比了本文和国内外的相关研究.

### 6.1 测试覆盖准则

目前,针对 FSM 的测试覆盖准则包括状态覆盖、迁移覆盖、完全谓词覆盖、迁移对覆盖、$n$ 迁移对覆盖、完全序列测试[8]和基于正则表达式的测试方法等.

Huang[4]提出分枝覆盖思想,即测试覆盖 FSM 中的每一条边,实际上是 FSM 的迁移覆盖.在测试时,该方法能够发现迁移中的错误,但并不能发现迁移组合中的错误.Pimont 等人[6]提出选择对覆盖,即考虑状态的输入条件和输出条件的组合,但这种方法不能发现选择对组合中的错误.Chow[3]提出了 W 方法,即将所有状态的选择对连接起来生成一棵树,从树的根到叶子节点的路径为测试用例,但在由该方法产生的测试用例中可能含有大量冗余,造成软件测试成本的增加.Howden[5]提出的内部边界覆盖,即覆盖 FSM 中的每一个状态.该方法的实质是图的点覆盖,由于可能存在边未被测试,因此该方法存在测试不完全现象.Fujiwara 等人[9]提出 Wp 方法,该方法从某个状态开始,产生相邻状态的迁移集合.Wp 方法采用集合的方式进行描述,第 $n$ 个状态的测试覆盖集合 $H_n=R_n\cup H_{n-1}$,其中,$R_n$ 表示状态 $R$ 到其他状态的序列,$H_{n-1}$ 表示 $n-1$ 个状态的测试覆盖集合,因此本质上与 $N$ 选择对集合覆盖一致.由于 Wp 方法在状态数目较多时产生的测试输入序列数目过多,从而影响了使用效率.为了提高 Wp 方法的效率,张涌等人[12]提出了 R-Wp 方法,采用的是减少额外状态的方法,以达到减少测试输入序列的目的.在 T 方法[10]中,测试输入序列对应规格说明中的状态迁移是随机产生的,直到所有的迁移都被覆盖.该方法的缺点是测试输入序列中存在大量的冗余.另外,其检错能力也较差.Alfred 等人[11]提出了 UIO(unique input/output sequence)方法,该方法首先从有限状态机的每一个状态得到一个识别序列,称为单一输入/输出序列,识别序列可以区分每一个状态,根据识别序列构造测试输入序列.但事实上,并不是所有的 FSM 都存在 UIO,如果 FSM 不存在 UIO,则无法构造测试输入序列.訾小超等人[20]提出了利用正则表达式描述隐含的信息的方法,这种方法能够找出 FSM 中的所有信息,明确状态迁移的细节,但会带来状态组合爆炸问题.张涌和钱乐秋等人[19]提出一种基于 EFSM 测试的测试数据自动选取方法,该方法将状态迁移中包含的表达式转化为正则表达式产生测试用例,但测试用例中存在大量冗余.

对比国内外的研究,本文提出的最小测试成本迁移覆盖准则有如下几点不同:

1) 针对的对象不同.最小测试成本迁移覆盖准则只针对 DFSM,并非所有的 FSM.
2) 测试准则的意图不同.最小测试成本迁移覆盖准则是为实现用最小的代价测试完 FSM 中的每一条迁移.在此基础上,可以实现本文在引言中提出的采用两个测试阶段实现测试成本和测试效率的平衡问题.
3) 构造思想不同.最小测试成本迁移覆盖准则是建立在集合论基础上的,拥有严谨的数学推理和证明,保证了该测试准则的完备性和可靠性.

### 6.2 组合爆炸问题

在由 FSM 模型导出的测试用例集合中可能会出现状态组合爆炸问题,为此,国内外的学者进行了大量的相关研究.Andrews 和 Jeff[13]提出了模型分层思想.这种思想将顶层的 FSM 分成不同的组(cluster),再将组细分为更详细的底层 FSM,由底层 FSM 产生测试用例进行测试.目前,这种方法主要运用在 Web 应用的测试中.

李华伟等人[21]根据电路输出信号的转换条件划分一个行为阶段聚类,利用 FSM 的行为阶段聚类产生测试序列测试电路门级固定型故障的诊断,阶段聚类的方法能够有效地解决状态组合爆炸问题.张涌等人[12]提出的 R-Wp 方法可在状态数目较大时产生相对较少的测试输入序列数目.Hierons 和 Ural[22]提出了对测试输入序列长度进行优化的方法,以减少测试序列的长度.随后,Hierons 等人[23]改进了区分序列,提出自适应区分序列来构造一棵决策树,有效地减少了测试序列的长度.王之梁等人[24]提出了 FSM 的全局变迁和局部变迁,用全局变迁来表示局部变迁,该方法的实质是 Jeff 的分层思想的逆过程.徐红等人[25]使用函数最小化方法自动产生 EFSM 的测试数据,该方法虽然能够产生最小的测试数据,但函数最小化方法是一种简单的搜索算法,每次只考虑一个约束条件,求出满足该约束条件的变量值后,再从当前变量值的基础上搜索满足下一个约束条件的变量值,因此可能需要进行多次回溯.徐宝文等人[26]通过测试需求约简的方法来获得规模较小的测试用例集,但该方法主要应用在回归测试当中.

与以往研究不同,本文的方法并不考虑FSM模型的分解或组合,而是从集合覆盖的角度出发,获取DFSM的最小测试成本迁移覆盖.经过有效转换后,获得的测试用例集合中冗余少且测试用例数目最少.因此,本文的方法同样避免了在测试用例生成过程中状态组合爆炸问题.

## 7 总结及未来工作

测试是为了发现软件中的错误,但现有的任何测试覆盖准则都不能保证发现软件中的所有错误,盲目增加测试用例并不是一种发现软件错误的好方法.实际测试时,软件中的错误往往集中在某个或几个模块中,若能花费最小的代价测试 DFSM 中所有的迁移,那么就能针对软件出错的部分集中进行测试.这种测试思想既能减少测试成本,又强调了对程序出错部分的重点测试,因此,该方法是解决测试成本和测试效率平衡问题的有效方式.为此,依据 DFSM 模型的相关理论和集合的覆盖理论,本文提出了针对 DFSM 模型的最小测试成本迁移覆盖准则.该准则借鉴了集合的覆盖及划分思想,通过获得 DFSM 的最小成本覆盖方式得到一个测试用例集合.由于该集合覆盖了 DFSM 中所有的迁移,且集合中任意两条迁移序列都相互独立,因此,该集合能够花费最小的代价覆盖 DFSM 中所有的迁移.

本文的贡献可以概括为:

1) 提出了两个阶段的测试方法,为解决测试成本和测试效率平衡问题提供了一种新思路;
2) 提出了最小测试成本迁移覆盖准则,给出了 DFSM 的最小测试成本迁移覆盖存在的必要性和充分性条件;
3) 设计了优化迁移覆盖算法和最小测试成本迁移覆盖算法.

在对软件需求进行建模时,有时FSM模型的迁移之间存在约束关系[13,27].文献[13]建议,在建模阶段,通过增加状态的方式,将带约束关系的FSM模型转换成无约束的FSM模型.而文献[27]提出了几种启发式算法寻找线性依赖中的约束关系.将来,我们可将这些算法思想应用于本文的方法,生成带迁移约束的测试用例集合.下一步工作的重点是,针对本文提出的第 2 个阶段测试,设计有效的测试方法.实际测试时,由于模型非常复杂且可能包含多个嵌套循环,因此即使发现系统与模型中的某个状态不一致,也很难确定系统的具体出错位置及出错来源.为此,我们将会研究复杂模型的分解及由分解后的子模型确定系统的错误位置和错误来源的相关技术.

**References**:

[1] Bertolino A. Software testing research: Achievements, challenges, dreams. In: Briand LC, Wolf AL, eds. Proc. of the Future of Software Engineering (FOSE 2007). Minneapolis: IEEE Computer Society Press, 2007. 85−103. [doi: 10.1109/FOSE.2007.25]

[2] Hopcroft JE, Motwani R, Ullman JD. Introduction to Automata Theory, Languages, and Computation. 2nd ed., Pearson Education Inc., 2001. 37−81. [doi: 10.1145/568438.568455]

[3] Chow TS. Testing software designs modeled by finite-state machines. IEEE Trans. on Software Engineering, 1978, SE-4(3):178−187. [doi: 10.1109/TSE.1978.231496]

[4] Huang JC. An approach to program testing. ACM Computer Survey, 1975,7(3):113−128. [doi: 10.1145/356651.356652]

[5] Howden WE. Methodology for the generation of program test data. IEEE Trans. on Computers, 1975,C-24(5):554−560. [doi: 10.1109/ T-C.1975.224259]

[6] Pimont S, Rault JC. A software reliability assessment based on a structural and behavioral analysis of programs. In: Proc. of the 2nd Int'l Conf. on Software Engineering. San Francisco: IEEE Computer Society Press, 1976. 486−491. http://www.informatik.uni-trier.de/~ley/db/conf/icse/icse76.html

[7] Ammann P, Offutt J. Introduction to Software Testing. Cambridge: Cambridge University Press, 2008. 27−103.

[8] Offutt J, Liu SY, Abdurazik A, Ammann P. Generating test data from state-based specifications. Software Testing, Verification and Reliability, 2003,13(1):25−53. [doi: 10.1002/stvr.264]

[9] Fujiwara S, von Bochmann G, Khendek F, Amalou M, Ghedamsi A. Test selection based on finite state models. IEEE Trans. on Software Engineering, 1991, 17(6):591−603. [doi: 10.1109/32.87284]

[10] Sidhu DP, Leung TK. Formal methods for protocol testing: A detailed study. IEEE Trans. on Software Engineering, 1989,15(4): 413−426. [doi: 10.1109/32.16602]

[11] Aho AV, Dahbura AT, Lee D, Uyar MU. An optimization technique for protocol conformance test sequence generation based on UIO sequence and rural Chinese postman tour. IEEE Trans. on Communications, 1991,39(11):1604−1615. [doi: 10.1109/ 26.111442]

[12] Zhang Y, Qian LQ, Wang YF. Test sequences selection based on deterministic finite-state machines. Journal of Computer Research and Development, 2002,39(9):1144−1150 (in Chinese with English abstract).

[13] Andrews AA, Offutt J, Alexander RT. Testing Web applications by modeling with FSMs. Software and Systems Modeling, 2005,4(3): 326−345. [doi: 10.1007/s10270-004-0077-7]

[14] Miao HK, Liu P, Mei J, Zeng HW. A new approach to automated redundancy reduction for test sequences. In: Proc. of the 15th Pacific Rim Int'l Symp. on Dependable Computing (PRDC 2009). Shanghai: IEEE Computer Society Press, 2009. 93−98. http://www.informatik.uni-trier.de/~ley/db/conf/prdc/prdc2009.html [doi: 10.1109/ PRDC.2009.23]

[15] Miao HK, Li G, Zhu GM. Software Engineering Language—Z. Shanghai: Scientific and Technical Documents Publishing House, 1999. 50−51, 105−112 (in Chinese).

[16] Duan LH, Chen J. Reducing test sequence length using invertible sequences. In: Proc. of the 9th Int'l Conf. on Formal Engineering Methods (ICFEM 2007). Boca Raton: IEEE Computer Society Press, 2007. 171−190. [doi: 10.1007/978-3-540-76650-6_11]

[17] Qian ZS, Miao HK, Zeng HW. A practical Web testing model for Web application testing. In: Yetongnon K, Chbeir R, Dipanda A, eds. Proc. of the 3rd Int'l Conf. on Signal-Image Technology & Internet-Based Systems (SITIS 2007). Shanghai: IEEE Computer Society Press, 2007. 404−411. [doi: 10.1109/SITIS.2007.16]

[18] Ural H, Wu XL, Zhang F. On minimizing the lengths of checking sequences. IEEE Trans. on Computers, 1997,46(1):93−99. [doi: 10.1109/12.559807]

[19] Zhang Y, Qian LQ, Wang YF. Automatic testing data generation in the testing based on EFSM. Chinese Journal of Computers, 2003,26(10):1295−1303 (in Chinese with English abstract).

[20] Zi XC, Yao LH, Li L. A state-based approach to information flow analysis. Chinese Journal of Computers, 2006,29(8):1460−1467 (in Chinese with English abstract).

[21] Li HW, Min YH, Li ZC. Clustering of behavioral phases in FSMs and its applications to VLSI test. Science in China (Series F—Information Sciences), 2002,45(6):462−478. [doi: 10.1360/02yf9040]

[22] Hierons RM, Ural H. Optimizing the length of checking sequences. IEEE Trans. on Computers, 2006,55(5):618−629. [doi: 10.1.1.108.2520]

[23] Hierons RM, Jourdan GV, Ural H, Yenigun H. Using adaptive distinguishing sequences in checking sequence constructions. In: Wainwright RL, Haddad H, eds. Proc. of the 23rd Annual ACM Symp. on Applied Computing (SAC 2008). IEEE Computer Society Press, 2008. 682−687. [doi: 10.1145/1363686.1363850]

[24] Wang ZL, Wu JP, Yin X. Protocol interoperability test generation based on communicating multi-port FSMs. Chinese Journal of Computers, 2006,29(11):1909−1919 (in Chinese with English abstract).

[25] Xi HY, Xu H, Gao ZY. ADA software test case generation tool. Journal of Software, 1997,8(4):297−302 (in Chinese with English abstract). http://www.jos.org.cn/ch/reader/view_abstract.aspx?flag=1&file_no=19970409&journal_id=jos

[26] Zhang XF, Xu BW, Nie CH, Shi L. An approach for optimizing test suite based on testing requirement reduction. Journal of Software, 2007,18(4):821−831 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/821.htm [doi: 10.1360/jos180821]

[27] Jun Y, Zhang J, Xu ZX. Finding relations among linear constraints. Lecture Notes in Computer Science, Vol.4120, Heidelberg: Springer-Verlag, 2006. 226−240. [doi: 10.1007/11856290_20]

**附中文参考文献:**

[12] 张涌,钱乐秋,王渊峰.基于确定有限状态机的测试输入序列选取.计算机研究与发展,2002,39(9):1144−1150.

[15] 缪淮扣,李刚,朱关铭.软件工程语言-Z.上海:上海科学技术文献出版社,1999. 50−51, 105−112.

[19] 张涌,钱乐秋,王渊峰.基于扩展有限状态机测试中测试输入数据自动选取的研究.计算机学报,2003,26(10):1295−1303.

[20] 訾小超,姚立红,李斓.一种基于有限状态机的隐含信息流分析方法.计算机学报,2006,29(8):1460−1467.

[24] 王之梁,吴建平,尹霞.基于通信多端口有限状态机的协议互操作性测试生成研究.计算机学报,2006,29(11):1909−1919.

[25] 奚红宇,徐红,高仲仪.Ada 软件测试用例生成工具.软件学报,1997,8(4):297−302. http://www.jos.org.cn/ch/reader/view_abstract.aspx?flag=1&file_no=19970409&journal_id=jos

[26] 章晓芳,徐宝文,聂长海,史亮.一种基于测试需求约简的测试用例集优化方法.软件学报,2007,18(4):821−831. http://www.jos.org.cn/1000-9825/18/821.htm [doi: 10.1360/jos180821]

**刘攀**(1976－),男,江西南昌人,博士,主要研究领域为软件工程,软件形式方法,软件测试,算法设计与 Web 开发.

**曾红卫**(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件形式方法,软件验证与软件测试.

**缪淮扣**(1953－),男,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件形式方法.

**梅佳**(1982－),男,博士,主要研究领域为软件形式方法,软件验证.