基于PEFSM行为模型的黑盒测试用例生成方法

梁浩然[1] 周宽久[1,2] 崔 凯[1] 潘 杰[1] 侯 刚[1]
(大连理工大学软件学院 大连 116620)[1] (大连理工大学软件评测中心 大连 116620)[2]

摘 要 随着计算机软件在医疗、航天、金融等领域的广泛应用,人们对软件系统可靠性的要求越来越严格。软件测试是保证软件安全可靠的有效手段,测试用例的优劣会直接影响测试效果及测试成本。针对嵌入式系统黑盒测试问题,提出了基于概率扩展有限状态机(PEFSM)行为模型的测试用例生成方法,通过两个假设给出了该方法的适用场景,设计了正则表达式转化和展开算法,并将该方法应用于Android智能电视的黑盒测试。该方法的特点是:1)根据用户对待测系统各类操作的使用频率信息,优先测试用户常用操作,从而缩减测试用例的数量和长度;2)可人为指定测试用例的初始状态和结束状态,设置闭包循环次数和迁移之间的等待时长,从而保证测试方法的灵活性和适用性。对比实验结果表明,该方法能够降低软件测试成本,提高测试用例的错误探测效率。

关键词 可信验证,PEFSM,正则表达式,自动化测试
中图法分类号 TP306 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.050

Black-box Test Case Generation Method Based on PEFSM Behavioral Model

LIANG Hao-ran[1] ZHOU Kuan-jiu[1,2] CUI Kai[1] PAN Jie[1] HOU Gang[1]
(School of Software, Dalian University of Technology, Dalian 116620, China)[1]
(Software Evaluation & Test Center, Dalian University of Technology, Dalian 116620, China)[2]

Abstract With the wide application of computer software in medicine, spaceflight, finance and other fields, people proposed stricter requirements for the reliability of the software system. Software testing is an effective method to ensure the security and reliability of software, whereas the quality of test cases will directly influence the testing efficiency and cost. To solve black-box testing problems about embedded systems, a test case generation method based on PEFSM (Probabilistic Extended Finite State Machine) behavioral model was proposed. The applicable scenarios of this method are illuminated with two assumptions. The algorithms for converting and unwrapping RE (regular expression) are designed, and this method is applied to conduct black-box testing towards a smart television based on Android operating system. The features of this method are as follows: 1) users' common operations on IUT (Implementation Under Test) can be tested preferentially by utilizing different frequencies of various operations. In this way, it is possible to reduce the number and length of the test cases. 2) testers can specify the initial and final states of test cases. In addition, they can also set cyclic numbers of closures and the waiting time between transitions. Therefore, this method makes the testing flexible and applicable. The experimental results show that this method can not only reduce the software testing cost, but also improve the error-detecting efficiency of test cases.

Keywords Credible verification, PEFSM, Regular expression, Automated testing

1 引言

随着计算机软件在医疗、航天、金融等领域的广泛应用,人们对软件系统的可靠性和安全性的要求越来越高。软件测试是保证软件可靠性的手段之一,但在测试过程中进行的重复性工作将花费大量的研发成本和时间。因此,通过软件自动化测试用例的生成和测试过程来提高软件测试效率,降低测试成本,是软件测试未来的发展趋势[1]。

有限状态机(Finite State Machine, FSM)[2-4]是一种对系统建模的常用方法,被广泛应用于对软件需求模型的描述。基于FSM的软件测试是一种模型驱动的测试用例生成方法。首先根据软件需求构建FSM模型,并在此基础上生成测试用例,然后在待测系统(Implementation Under Test, IUT)[5]中执行测试用例,检验实际运行结果是否满足软件需求说明。刘攀等[6]从测试成本、错误探测能力等方面对 W[7], W_p[5], UIO[8]等多种基于FSM的测试方法进行了评估和比较。

到稿日期:2016-03-17 返修日期:2016-07-13 本文受国家自然科学基金(61402073)资助。
梁浩然(1993—),男,硕士生,主要研究方向为可信软件、软件测试;周宽久(1966—),男,博士,教授,主要研究方向为嵌入式系统工程、软件形式化验证、软件可靠性;崔 凯(1983—),男,博士生,主要研究方向为可信软件、软件形式化验证;潘 杰(1992—),男,硕士生,主要研究方向为可信软件、软件测试;侯 刚(1982—),男,博士生,讲师,主要研究方向为软件形式化验证,E-mail:hg.dut@163.com(通信作者)。

Ammann 和 Offutt[9] 讨论了根据软件源码或需求说明构造 FSM 的方法，并提出状态覆盖、变迁覆盖和边对覆盖标准，但没有给出针对 FSM 的测试用例生成方法及具体算法实现。

在 FSM 的基础上，扩展有限状态机（Extended Finite State Machine，EFSM）[10-14] 被提出。EFSM 引入了上下文变量和迁移执行条件等，带有记忆功能，比 FSM 具有更强的建模能力，因此常用于复杂系统的建模和测试用例的生成。Huang 等[15] 利用 EFSM 对软件进行数据流和控制流建模，提出了一种通过可达性分析生成可执行测试序列的方法，但该方法复杂度较高。Zhang 等[16] 提出了利用启发式算法自动生成 EFSM 测试用例的方法，但该方法需要进一步优化以提高算法的执行效率。

RE[17] 是与 FSM 等价的一种符号化的模型描述方法，能够充分描述 FSM 中的循环等信息，而且更易于计算机处理，因此也可以应用于测试用例的生成。Liu 等[18] 提出了一个描述软件行为的正则表达式代数系统，但只是以实例分析的方式介绍测试用例生成方法，没有给出通用的形式化算法。Belli 和 Grosspietsch[19] 结合正则表达式和谓词/迁移网络实现对指定测试用例的容错，但未给出测试用例的生成方法。

由于实际用户对软件系统中各种功能的使用频率不同，因此不同功能中存在的错误也具有不同的重要性。在进行软件测试时可以利用这种用户偏好信息减少测试用例，尽可能地降低测试成本。本文提出了一种指向性的黑盒测试用例生成方法，模拟用户对软件的实际操作，根据用户在实际使用场景中对不同功能的不同使用频率按概率生成测试用例。通过该方法得到的测试用例更贴合软件实际运行环境，在此基础上进行的测试也更加具有说服性，能够在保证一定错误探测能力的同时尽可能地降低测试成本。

本文第 2 节介绍了相关定义及概率扩展有限状态机（Probabilistic Extended Finite State Machine，PEFSM）的构造方法；第 3 节给出了正则表达式（Regular Expression，RE）转化策略以及添加扩展项的方法；第 4 节介绍了 RE 的展开算法及基于 Markov 链的概率选择算法；第 5 节对实验结果进行了分析，证明了该方法的有效性；最后总结全文并展望未来工作。

2　概率扩展状态机的构造策略

2.1　相关定义

下面给出一些关于 PEFSM 的定义，以便理解本文提出的黑盒测试用例生成方法。

定义 1　PEFSM M 被定义为一个八元组 $M=(I,O,S,s_0,V,P,A,T)$。其中，I，O 和 S 分别表示有限非空的输入符号集、输出符号集和状态集合，s_0 表示初始状态，V 表示上下文变量集，P 表示由变量构成的谓词表达式集合，A 表示迁移动作集，T 表示带概率的有限非空状态迁移集。$\forall t\in T, t=(s_i,i,p,a,o,s_j,b)$，其中 $s_i,s_j\in S$，分别称为迁移 t 的迁出状态和迁入状态，用 $t.Start$ 和 $t.End$ 表示；$i\in I$ 和 $o\in O$ 分别代表迁移的输入符号和输出符号，用 $t.Input$ 和 $t.Output$ 表示；$p\in P$ 表示迁移所需满足的谓词表达式，用 $t.Predicate$ 表示；$a\in A$ 表示迁移发生时对上下文变量的操作，用 $t.Acion$ 表示；b 表示在从状态 s_i 流出迁移中的输入符号组成集合 I' 中选择 i 触发的概率，用 $t.Probability$ 表示；并且在任意状态配置 $s_i(v_{i1},v_{i2},\cdots,v_{in})$ 下，所有可执行迁移的概率之和为 1，即 $\Sigma_k b_k=1$。

定义 2　若 PEFSM 运行的当前状态为 s_i，且上下文变量集中的变量 $v_1,v_2,\cdots,v_n$ 分别取值为 $v_{i1},v_{i2},\cdots,v_{in}$，则可称 PEFSM 当前的状态配置为 $s_i(v_{i1},v_{i2},\cdots,v_{in})$，简写为 SC_i。状态机最开始的状态配置被称为初始状态配置，其中 $s_i=s_0$，且 $v_{i1},v_{i2},\cdots,v_{in}$ 被称为初始变量值。

定义 3　若迁移 $t=(s_i,i,p,a,o,s_j)$ 在某状态配置 $s_i(v_{i1},v_{i2},\cdots,v_{in})$ 下满足 p 为真，则称 t 是关于状态配置 SC_i 的一条可执行迁移。其中，$s_i(v_{i1},v_{i2},\cdots,v_{in})$ 被称为 t 的迁出状态配置。假设执行动作 a 后上下文变量 $v_1,v_2,\cdots,v_n$ 分别取值为 $v_{j1},v_{j2},\cdots,v_{jn}$，则 $s_j(v_{j1},v_{j2},\cdots,v_{jn})$ 被称为 t 的迁入状态配置。

为了确定 PEFSM 行为模型建模的适用场景，本文提出两个基本假设。

假设 1　在任意状态配置 $s_i(v_{i1},v_{i2},\cdots,v_{in})$ 下，由所有从状态 s_i 流出迁移中的输入符号组成集合 I，对于 $\forall i\in I$，有且只有一条可执行迁移 t，使得 t 的输入符号为 i。

假设 2　在状态 s_i 相同的任意两个状态配置下，所有从状态 s_i 流出且输入符号相同的迁移的发生概率相同。

假设 1 保证了系统在任意状态配置下，以流出迁移中出现的任意输入符号作为输入必定会触发某一条迁移，也被称为确定性迁移假设；假设 2 保证了迁移概率只与迁出状态和输入符号有关，与谓词表达式无关，也被称为同等概率假设。

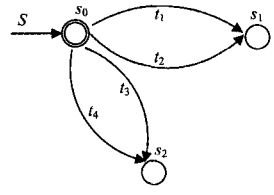

图 1　示例状态机

以图 1 中的示例状态机为例，若其满足假设 1 且 s_0 状态的流出迁移 t_1,t_2,t_3,t_4 分别对应的输入符号为“AB”，“AB”，“BC”，“CD”，则输入“AB”时，t_1,t_2 中有且只有一条迁移被触发；输入“BC”时，t_3 一定会被触发。若其满足假设 2，则 t_1 和 t_2 始终具有相同的发生概率。

2.2　概率扩展状态机的构造

在黑盒测试中，PEFSM 的构造要根据软件需求说明进行建模。首先根据需求说明抽象出状态、输入、变量和输出集合，然后确定每条迁移的条件和动作，通过统计分析得到每条迁移对应的概率，最后绘制出与 PEFSM 等价的状态迁移图。

智能电视是近几年出现的比较流行的嵌入式设备，由于当前技术还不够成熟，在设计和实现中容易产生许多错误，亟需通过软件测试等方法保证系统的正确性。本文以某品牌基于 Android 操作系统的智能电视为黑盒测试对象，介绍基于 PEFSM 行为模型的构造以及黑盒测试用例生成方法。

根据系统需求说明，可将光标位置抽象为操作系统的状态，将按键操作抽象为输入符号，将光标移动抽象为状态间的

迁移。通过分析发现，某个状态下可触发光标移动的按键输入在任意状态配置下都有效，满足确定性迁移假设；各按键的概率只与当前光标位置有关，满足同等概率假设。因此，智能电视系统可利用 PEFSM 进行建模。

图 2 示出了智能电视某用户界面的示意图，为了简洁起见，只列出了部分与状态机构造有关的文字，其中矩形对应着光标可能移动到的位置，最下方 8 个矩形对应着电视启动后的 8 个主界面，通过移动光标到不同矩形可以实现主界面间的切换。各主界面包含的功能选项显示在整个用户界面的中间区域。例如，图 2 中的“应用商店”、“卸载应用”等是隶属于“应用”主界面的功能选项。上方区域是“工具箱”窗口，包括“设置”等 7 个选项，当点击遥控器的“工具箱”按键时，在任意界面都会弹出该窗口。

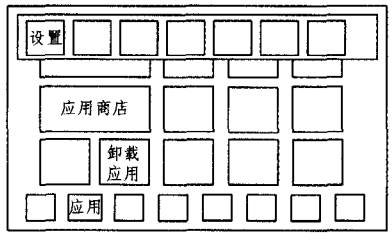

图 2 智能电视界面示意图

基于图 2，可以构造出如图 3 所示的 PEFSM。其中，B100 对应“应用商店”选项，既是初始状态又是唯一的终止状态，B200 对应“卸载应用”选项，b000 对应“应用”选项，I000 对应“设置”选项，输入符号为遥控器键值，上下文变量集合 $V=\{BMem, BbSelect\}$，初始状态配置为 B100(“B200”，“b”)，迁移 t_1-t_8 的含义如表 1 所列。

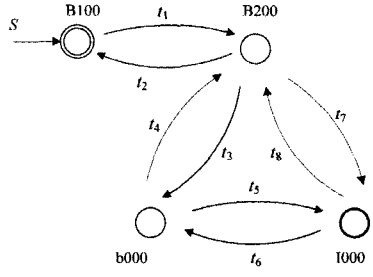

图 3 “应用”主界面对应状态机

表 1 迁移含义表

迁移	输入	条件	动作	输出	概率
t_1	BD	True	ε	ε	1.0
t_2	BC	True	ε	ε	0.3
t_3	BD	True	BMem=“B200”	ε	0.6
t_4	BC	BMem==“B200”		ε	0.7
t_5	B2	True	BbSelect=“b”	ε	0.3
t_6	B9	BbSelect==“b”	ε	ε	1.0
t_7	B2	True	BMem=“B200”; BbSelect=“B”	ε	0.1
t_8	B9	BbSelect==“B” &&BMem==“B200”	ε	ε	1.0

当光标从中间区域离开时，系统会记录光标在中间区域停留的最后一个位置，并将它作为当光标重新回到中间区域时的默认位置。因此，用 BMem 变量记录这个位置所对应状态的名称。当光标位于中间或下方区域时，若按下“工具箱”键，光标会跳转到上方区域的“设置”位置。此时若按下“返回”键，光标会回到原位置，因此需要记录光标之前所处区域，用 BbSelect 变量存储，“B”表示光标原位置在中间区域，“b”表示光标原位置在下方区域。

3 正则表达式的转化策略

3.1 正则表达式的扩充

正则表达式是与 FSM 等价的一种描述模型，其基本组成元素是输入符号，而与 PEFSM 等价的正则表达式除了输入符号还应该包括迁移条件、迁移动作、迁移概率等。本文将正则表达式进行扩充，将其基本组成元素定义为状态机中的迁移，并且增加闭包次数和迁移间等待时长两种扩展项，定义如下。

定义 4 给定一个 PEFSM $M=(I,O,S,s_0,V,P,A,T)$，则 $\forall t\in T$。

(1)$t=(s_i,i,p,a,o,s_j,b)$是扩充的正则表达式。

(2)$t\{T=m\}$是扩充的正则表达式，其中 $m\in N^+$，表示 t 中的迁移执行后等待 m 秒后再进行下一次迁移。$\{T=m\}$称为等待时长项。

(3)如果 r 和 s 都是扩充的正则表达式，则$(r\&s)$是扩充的正则表达式，表示连接关系；$(r|s)$是扩充的正则表达式，表示选择关系；(r^*)是扩充的正则表达式，表示克林闭包；$(s^*\{N=n\})$是扩充的正则表达式，表示指定 s 的循环重复次数为 n 次的克林闭包。其中，$\{N=n\}$称为闭包次数项。“*”的优先级大于“&”，“&”的优先级大于“|”，在不引起歧义的情况下可以省略圆括号。例如$(((a\&b)|c)|d)$可简写为 $a\&b|c|d$。

除特殊说明外，下文提到的正则表达式均为本节定义的扩充的正则表达式。

3.2 转化策略

对于 PEFSM M，初始状态为 s_0，终止状态集合为 F，则生成正则表达式的方法如下。

(1)去掉 M 中以 s_0 为起始点的不可达状态以及不可到达任意终止状态的状态，增加 X 状态和 Y 状态，从 X 状态引一条标记为 ε 的迁移到 s_0 状态，对 $\forall s\in F$，从 s 状态引一条标记为 ε 的迁移到 Y 状态。

(2)$\forall a,b\in S\cup\{X,Y\}$，若从 a 到 b 存在多条迁移，则将这些迁移的标记用“或”运算符连接起来作为一条从 a 到 b 迁移的标记，并删除其他迁移。

(3)$\forall a,b,c\in S\cup\{X,Y\}$，若从 a 到 b 存在迁移 t_1，从 b 到 c 存在迁移 t_2，不存在从 b 到 b 的迁移，则将这两条迁移删除，并添加一条标记为 $t_1\&t_2$ 的迁移。

(4)$\forall a,b,c\in S\cup\{X,Y\}$，若从 a 到 b 存在迁移 t_1，从 b 到 c 存在迁移 t_2，从 b 到 b 存在迁移 t_3，则将这 3 条迁移删除，并添加一条标记为 $t_1\&t_2^*\&t_2$ 的迁移。

(5)重复步骤(2)—步骤(4)，直到图中只包含 X 和 Y 状态，且两状态间最多只保留一条迁移为止。此条迁移上的标记则为所求的正则表达式。如果迁移不存在，则正则表达式为空。

图 4 示出了将 2.2 节中构造的 PEFSM 转化为正则表达

式的过程，首先增加状态 X 和 Y 以及 ε 迁移，然后依次去除状态 I000，b000，B200，最终只保留状态 X 和 Y，得到如下的正则表达式：

$(t_1 \& (t_7 \& t_8 \mid (t_3 \mid t_7 \& t_6) \& (t_5 \& t_6)^* \& (t_4 \mid t_5 \& t_8))^* \& t_2)^*$

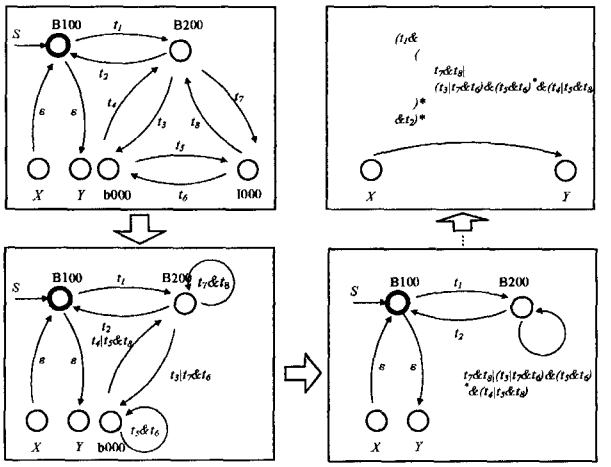

图 4　概率扩展 RE 转化示意图

3.3　添加扩展项

在正则表达式转化完成后，测试人员可以根据实际需要在正则表达式的合适位置添加扩展项，从而生成满足测试需求的测试用例，也体现出该测试用例生成方法的灵活性和适用性。例如，对 3.2 节中的正则表达进行式扩展后可得到：

$(t_1 \& (t_7 \& t_8 \{T=3\} \mid (t_3 \mid t_7 \& t_6) \& (t_5 \& t_6)^* \& (t_4 \mid t_5 \& t_8))^* \{N=2\} \& t_2)^*$

其中，$\{T=3\}$ 是等待时长项，表示在 t_8 迁移执行结束后等待 3 秒后再继续执行下一条迁移，等待时长项会在最终生成的测试用例得以保留；在测试过程中当系统执行某些耗时操作时，黑盒测试软件可以根据等待时长项确定下一次输入测试数据的时间，从而实现完全的自动化测试。$\{N=2\}$ 是闭包循环次数项，表示对 $(t_7 \& t_8 \{T=3\} \mid (t_3 \mid t_7 \& t_6) \& (t_5 \& t_6)^* \& (t_4 \mid t_5 \& t_8))^*$ 这个克林闭包中的子表达式要循环展开两次。根据测试人员设定的闭包次数项可以确定克林闭包的循环次数，从而在测试用例生成阶段将其展开，有针对性地对部分迁移进行重复测试。

4　测试用例的概率生成策略

测试用例的生成分为 3 个步骤。首先根据扩充的正则表达式进行展开，得到等价的不包含闭包和括号的与或式；然后利用 Markov 链计算与或式中每条与子式的概率，根据概率按比例选择与子式；最后经过处理选中的与子式得到最终的测试用例。

4.1　正则表达式的展开

在介绍正则表达式的展开算法之前，首先引入与或式、与子式、外层与子项和外层或子项的定义。

定义 5　与或式是形如 $t_1 \& t_2 \& \cdots \& t_i \mid t_{i+1} \& t_{i+2} \& t_2 \& \cdots \& t_j \mid \cdots \mid t_{j+1} \& t_{j+2} \& \cdots \& t_k$ 的只包含“与”、“或”两种运算符的正则表达式，其中 $t_1 \cdots t_k$ 均为迁移，并且 $t_1 \& t_2 \& \cdots \& t_i$，…，$t_{j+1} \& t_{j+2} \& \cdots \& t_k$ 均被称为该式的与子式。

定义 6　若 $r,s,\cdots,t$ 为正则表达式，则对于形如 $(r) \& (s) \& \cdots \& (t)$ 的正则式，$r,s,\cdots,t$ 均称为 $(r) \& (s) \& \cdots \& (t)$ 的外层与子项。

定义 7　若 $r,s,\cdots,t$ 为正则表达式，则对于形如 $(r) \mid (s) \mid \cdots \mid (t)$ 的正则式，$r,s,\cdots,t$ 均称为 $(r) \mid (s) \mid \cdots \mid (t)$ 的外层或子项。

正则表达式的展开算法是通过函数递归调用来实现的，算法输入为概率扩展正则式，算法输出为展开后的与或式，其基本思想如下：

(1)预处理，去除正则表达式最外层的多余括号，并将其中的所有时间项 $\{T=\cdots\}$ 替换为 $\& TM(\cdots)$。

(2)若得到的正则表达式形如 $(r) \mid (s) \mid \cdots \mid (t)$，则对外层或子项 (r)，(s)，…，(t) 分别递归调用该算法处理，将结果通过“或”运算按顺序连接并返回。

(3)若得到的正则表达式形如 $(r) \& (s) \& \cdots \& (t)$，则对外层与子项 (r)，(s)，…，(t) 分别递归调用该算法进行处理，得到与或式 (r')，(s')，…，(t')，然后利用“与”运算对“或”运算的分配率对 $(r') \& (s') \& \cdots \& (t')$ 进行展开，得到与或式并返回。

(4)对于未指定闭包循环次数的闭包，递归调用该算法处理闭包内的子表达式，得到与或式 (r)。若与或式中包含 m 条与子式，则将 (r) 重复 m 次，用“与”运算符连接，然后利用“与”运算对“或”运算的分配率将其展开为与或式并返回。

(5)对于指定闭包循环次数为 n 的闭包，同样递归调用该算法处理闭包内的子表达式，得到与或式 (r)。若 (r) 中只包含一条与子式，则将该与子式重复 n 次，用“与”运算符连接并返回；否则，若与或式中包含 m 条与子式，则将 (r) 重复 n 次，用“与”运算符连接，然后利用“与”运算对“或”运算的分配率将其展开为与或式并返回。

(6)如果算法输入的正则表达式只包含一条迁移或只由 $TM(\cdots)$ 组成，并不包含其他运算符，则直接返回该迁移或 $TM(\cdots)$。

步骤(1)替换时间项的目的是方便对正则表达式进行处理。步骤(4)将与子式的个数设定为闭包循环次数，是因为与子式个数可以反映闭包的复杂度。对于越复杂的闭包，应该赋予越大的循环次数，从而提高迁移覆盖率。步骤(3)、步骤(5)均利用“与”运算对“或”运算的分配率展开相与的与或式，如算法 1 所示。

算法 1　利用分配率展开相与的与或式

输入：与或式 $R_1, R_2, \cdots, R_n$

输出：与或式 E

```
1. Initialize List A←{0,0,…,0} whose size is n
2. Initialize List B1=B2=…=Bn←∅
3. Initialize String D=E←ε
4. for each Ri do
5.    for each and-minor s which belongs to Ri do
6.        Bi. Append(s)
7.    end for
8. end for
9. while true
10.   D←ε
11.   for i←0 to n do
12.       D←D+"&"+ Bi[A[i]]
```

```
13.   end for
14.   E←E +"|"+ D
15.   j←0
16. CFlag←true
17.   while CFlag do
18.     if j=n then
19.         return E
20.      end if
21.      if A[i]+1 < length[Bi] then
22.         CFlag←false
23.         A[i]←A[i]+1
24.     else then
25.         j←j+1
26.     end if
27.   end while
28. end while
```

算法 1 首先将每个与或式 R_i 中的与子式保存到列表 B_i 中，$A[i]$表示当前选择的与子式在列表 B_i 中的下标。通过各 $A[i]$从 0 到 length[B_i]的遍历，实现各与或式中与子式的排列组合，通过"|"运算符连接起来组成 E 并返回。

以 3.3 节中的概率扩展正则式为例，$(t_7 \& t_8\{T=3\} | (t_3 | t_7 \& t_6) \& (t_5 \& t_6)^* \& (t_4 | t_5 \& t_8))^* \{N=2\}$闭包内的子表达式经处理后得到 $t_7 \& t_8 \& TM(3) | t_3 \& t_5 \& t_6 \& t_5 \& t_8 | t_7 \& t_6 \& t_5 \& t_6 \& t_4 | t_7 \& t_6 \& t_5 \& t_6 \& t_4 | t_3 \& t_5 \& t_6 \& t_4$，共由 5 个与子式组成。由于指定了闭包次数，此时要对整个闭包执行步骤(5)的处理过程，得到由 $5^2=25$ 条与子式组成的与或式。因篇幅所限，表 2 只列出了其中 5 条与子式。

表 2 与子式及其对应概率

序号	与子式	选择概率	归一概率
1	$t_1 \& t_7 \& t_8 \& TM(3) \& t_7 \& t_8 \& TM(3) \& t_2$	0.0030000	0.194
2	$t_1 \& t_7 \& t_8 \& TM(3) \& t_3 \& t_5 \& t_6 \& t_4 \& t_2$	0.0037800	0.254
3	$t_1 \& t_3 \& t_5 \& t_6 \& t_5 \& t_8 \& t_7 \& t_8 \& TM(3) \& t_2$	0	0
4	$t_1 \& t_3 \& t_5 \& t_6 \& t_4 \& t_7 \& t_8 \& TM(3) \& t_2$	0.0037800	0.254
5	$t_1 \& t_3 \& t_5 \& t_6 \& t_4 \& t_3 \& t_5 \& t_6 \& t_4 \& t_2$	0.0047628	0.299

4.2 基于 Markov 链的概率选择

定义 8 对于随机序列$\{X_1, X_2, \cdots, X_n\}$，对 $\forall 1 \leqslant i \leqslant n$，$X_i$ 属于 $\Omega_q=\{\theta_1, \theta_2, \cdots, \theta_N\}$，若该随机序列在 $m+k$ 时刻所处的状态为 q_{m+k} 的概率只与它在 m 时刻的状态 q_m 有关，而与 m 时刻之前它所处的状态无关，即

$$P(X_{m+k}=q_{m+k} | X_m=q_m X_{m-1}=q_{m-1}, \cdots, X_1=q_1) = P(X_{m+k}=q_{m+k} | X_m=q_m) \quad (1)$$

则称$\{X_1, X_2, \cdots, X_n\}$ 为 Markov 链。其中 $q_1, q_2, \cdots, q_m, q_{m+k} \in \Omega_q$。

由于 PEFSM 中当前状态各迁出迁移中的概率项决定了下一个状态，即 $m+1$ 时刻所处的状态只与 m 时刻的状态有关，可将 PEFSM 运行的状态序列看成 Markov 链，利用 Markov 链的性质计算状态序列的概率。

对 PEFSM $M=(I, O, S, s_0, V, P, A, t)$，用 Markov 链建模，则 $\Omega_q=S, k=1$，对 $\forall t=(s_i, i, p, a, o, s_j, b)$，有：

$$P(X_{m+1}=s_j | X_m=s_i, X_{m-1}=q_{m-1}, \cdots, X_1=q_1) = P(X_{m+1}=s_j | X_m=s_i) = b \quad (2)$$

其中 $q_1, q_2, \cdots, q_{m-1} \in S$。对迁移 $t_1, t_2, \cdots, t_m$，如果满足 $\forall 1 \leqslant k \leqslant m-1$，$t_k=(s_{ki}, i_k, p_k, a_k, o_k, s_{kj}, b_k)$，有 $t_k.End=t_{k+1}.Start$，则有：

$$\begin{aligned} &P(X_{m+1}=s_{mj}, X_m=s_{mi}, X_{m-1}=s_{(m-1)i}, \cdots, X_1=s_{1i}) \\ &=P(X_{m+1}=s_{mj} | X_m=s_{mi}, X_{m-1}=s_{(m-1)i}, \cdots, X_1=s_{1i}) \\ &P(X_m=s_{mi} | X_{m-1}=s_{(m-1)i}, \cdots, X_1=s_{1i}) \cdots P(X_2=s_{2i} | X_1=s_{1i}) \\ &=P(X_{m+1}=s_{mj} | X_m=s_{mi}) P(X_m=s_{mi} | X_{m-1}=s_{(m-1)i}) \cdots P(X_2=s_{2i} | X_1=s_{1i}) \\ &=b_m b_{m-1} \cdots b_1 \end{aligned} \quad (3)$$

因此，可以通过计算迁移序列中概率的乘积得到执行此迁移序列的概率。在概率计算过程中需要注意只有当迁移条件满足时才将其概率参与计算，并且通过执行其动作对上下文变量进行修改，否则直接将此迁移序列的选择概率置为 0。基于 Markov 模型的与子式选择概率算法如算法 2 所示。

算法 2 计算与子式选择概率

```
输入：与子式 R1, R2, …, Rn 和上下文变量赋值语句集合 S
输出：选择概率 P1, P2, …, Pn
1. Initialize P1=P2=…=Pn=1.
2. for each s∈S do
3.    execute s//Initialize context variables
4. end for
5. for each Ri do
6.    for each transition t which belongs to Ri do
7.    if t.Predicate=true then
8.        for each a∈t.Action do
9.            execute a
10.       end for
11.       Pi←Pi * t.Probability
12.       else then
13.          Pi←0
14.             break
15.       end if
16.   end for
17. end for
18. return P1, P2, …, Pn
```

算法首先执行 S 中的每一条语句，实现对所有上下文变量的初始化，然后依次计算每个与子式的概率。对于与子式 R_i，从前往后依次处理 R_i 中的迁移 t。若 t 的谓词表达式为真，则执行 t 的动作集中的每一条赋值语句，并计算新的选择概率 P_i，最终得到所有的选择概率。表 2 中的各与子式经过计算后可以得到相应的选择概率，如表 2 中第 3 列所示。

将与子式的长度定义为其中包含的迁移个数，可以看出选择概率会受与子式长度的影响，例如若各迁移概率相等且小于 1，则长度越长，概率越小；长度越短，概率越大。为避免长度影响与子式的选择，通过以下公式计算归一概率，作为最终的与子式选择依据。

$$N_i=\frac{\sqrt[l_i]{C_i}}{\sum_{i=1}^{n}\sqrt[l_i]{C_i}} \quad (4)$$

其中，N_i 表示所求的归一概率，C_i 表示第 i 条与子式的选择概率，l_i 表示第 i 条与子式的长度。经过计算后，可得到如表 2 第 4 列所列的归一概率。

4.3 可执行测试用例生成

定义 9 可执行测试序列$\langle t_1, \cdots, t_n \rangle$是由可执行迁移组

成的有序列表，并且 $\forall i \in N$，若 $1 \leqslant i < n$，则 $t_i.End$ 的迁入状态配置与 $t_{i+1}.Start$ 的迁出状态配置完全相同。其中，t_1 的迁出状态配置被称为该测试序列的迁出状态配置，t_n 的迁入状态配置被称为该测试序列的迁入状态配置，$\langle t_1.Input, t_2.Input, \cdots, t_n.Input\rangle$ 被称为该测试序列的输入序列，$\langle t_1.Output, t_2.Output, \cdots, t_n.Output\rangle$ 被称为该测试序列的输出序列。

定义 10　若 PEFSM 的初始状态配置为 SC_0，存在一条以 SC_0 为迁出状态配置的可执行测试序列 S，则 S 的输入序列、输出序列和 SC_0 共同组成了该状态机的一条可执行测试用例。

若表 2 中按归一概率选择的正则表达式为第一条与子式，则输入序列〈BD，B2，B9，B2，B9，BC〉、输出序列〈ε，ε，ε，ε，ε，ε〉和初始状态配置 B100（"B200"，"b"）共同组成了一条可执行的测试用例，其中两次输入 B2 后都需要等待 3 秒。

5　实验分析

根据 PEFSM 行为模型的测试用例生成方法，设计和实现了针对 Android 智能电视操作系统的测试平台，并在该平台上完成实验。根据用户输入的迁移信息和初始配置信息，该平台可自动实现正则表达式的转化、展开和概率选择等过程，生成可执行的迁移序列。该平台还可以根据数据库中存储的按键映射关系，将迁移中的输入符号转化为 Android 系统可接收的键值码，利用 MonkeyRunner 工具模拟真实遥控器的按键操作，从而实现了测试自动化。

在待测系统中，用户访问越频繁的功能区域所存在的错误影响越大，重要性越高，因此可以认为能够发现这些错误的测试用例具有更强的错误探测能力。为了量化评估不同类型错误的重要性，本文将错误类型按照所在功能区域的使用频率大致分为高、中、低 3 类，并分别对应 3 个数值"3"，"2"，"1"，从而对错误的重要程度进行描述。实验以 2.2 节中构造的 PEFSM 模型作为测试对象，向待测系统中注入 6 个错误，如表 3 所列。

表 3　错误类型表

序号	涉及迁移		错误数	功能使用频率	重要性数值
1	t_3	t_4	2	高	3
2	t_2	t_5	2	中	2
3	t_8	t_7	2	低	1

本文参考文献[6]中测试成本的定义给出错误探测效率 R 的计算方法，如式(5)所示。其中，C_i 表示发现第 i 类型错误的个数，M_i 表示第 i 类型错误的重要度，S_q 表示测试用例的总个数，L_i 表示第 i 条测试用例的长度，α 是小于 1 的测试参数，用于控制测试用例长度和个数对测试成本的影响，根据假设，本文设定参数 α 为 0.2。

$$R=\frac{\sum_{i=1}^{n} C_i M_i}{\alpha S_q+(1-\alpha)\sum_{1\leqslant i\leqslant S_q} L_i} \tag{5}$$

实验将本文所提方法与状态覆盖、迁移覆盖方法进行了比较，实验结果如表 4 所列。从表 4 中可以看出，本文所提方法的测试用例总长度最短，探测错误数小于迁移覆盖，但总体上错误探测效率高于其他两种方法。这是因为该方法容易发现用户使用过程中出现频率高的错误，而这些错误的重要性也更高，从而提高了整体的错误探测效率。

表 4　实验结果比较

测试方法	用例个数	用例总长度	探测错误数	探测效率(R)
本文方法	1	10	4	1.220
状态覆盖	2	4	2	1.111
迁移覆盖	5	14	6	0.984

结束语　基于 FSM 等模型生成测试用例是软件测试中的常用方法，但现有的测试用例生成方法都无法保证能够发现软件中的所有错误，因此本文提出了一种更切合实际的测试思路，在保证一定错误探测能力的前提下尽可能地降低测试成本。该方法通过构建 PEFSM 行为模型，充分利用各功能模块的用户使用频率信息，按概率选择出测试用例，从而实现对重要性更高的用户常用功能进行集中测试，能够有效提高测试用例的错误探测效率。

下一步工作的重点是对文中提出的正则表达式转化和展开算法进行优化，降低计算复杂度，以及进一步提高该方法的适用性，研究针对大型复杂系统的层次建模和组合测试问题。

参 考 文 献

[1] BERTOLINO A. Software testing research: Achievements, challenges, dreams [C] // Future of Software Engineering. IEEE, 2007: 85-103.

[2] LEE D, YANNAKAKIS M. Principles and methods of testing finite state machines-a survey[J]. Proceedings of the IEEE, 1996, 84(8): 1090-1123.

[3] PRADHAN S N, CHOUDHURY P. Low power and high testable Finite State Machine synthesis [C] // 2015 International Conference and Workshop on Computing and Communication (IEMCON). Vancouver, Canada: IEEE Press, 2015: 1-5.

[4] GOMES CABRAL F, MOREIRA M V, DIENE O, et al. A Petri net diagnoser for discrete event systems modeled by finite state automata[J]. IEEE Transactions on Automatic Control, 2015, 60(1): 59-71.

[5] FUJIWARA S, BOCHMANN G V, KHENDEK F, et al. Test selection based on finite state models[J]. IEEE Transactions on Software Engineering, 1991, 17(6): 591-603.

[6] LIU P, MIAO H K, ZENG H W, et al. FSM-based testing: Theory, method and evaluation[J]. Chinese Journal of Computers, 2011, 34(6): 965-984. (in Chinese)
刘攀，缪淮扣，曾红卫，等. 基于 FSM 的测试理论，方法及评估[J]. 计算机学报，2011，34(6)：965-984.

[7] CHOW T S. Testing software design modeled by finite-state machines[J]. IEEE Transactions on Software Engineering, 1978 (3): 178-187.

[8] SABNANI K, DAHBURA A. A protocol test generation procedure[J]. Computer Networks and ISDN systems, 1988, 15(4): 285-297.

[9] AMMANN P, OFFUTT J. Introduction to software testing [M]. Cambridge, UK: Cambridge University Press, 2008.

[10] BOCHMANNAND G V, GECSEI J. A unified method for the specification and verification of protocols[C] // IFIP Congress, 1977. Toronto, Canada: IEEE, 1977: 229-234.

[11] PETRENKO A, BORODAY S, GROZ R. Confirming configura-

tions in EFSM testing[J]. IEEE Transactions on Software Engineering, 2004, 30(1): 29-42.

[12] DERDERIAN K, HIERONS R M, HARMAN M, et al. Estimating the feasibility of transition paths in extended finite state machines[J]. Automated Software Engineering, 2010, 17(1): 33-56.

[13] YANG R, CHEN Z, ZHANG Z, et al. EFSM-Based Test Case Generation: Sequence, Data, and Oracle[J]. International Journal of Software Engineering and Knowledge Engineering, 2015, 25(4): 633-667.

[14] WALKINSHAW N, TAYLOR R, DERRICK J. Inferring extended finite state machine models from software executions[J]. Empirical Software Engineering, 2016, 21(3): 811-853.

[15] HUANG C M, JANG M Y, LIN Y C. Executable EFSM-based data flow and control flow protocol test sequence generation using reachability analysis[J]. Journal of the Chinese Institute of Engineers, 1999, 22(5): 593-615.

[16] ZHANG J, YANG R, CHEN Z, et al. Automated EFSM-based test case generation with scatter search [C] // International Workshop on Automation of Software Test (AST). IEEE Press, 2012: 76-82.

[17] GAROFALAKIS M N, RASTOGI R, SHIM K. SPIRIT: Sequential pattern mining with regular expression constraints: VLDB, 1999[C] // Edinburgh. Scotland: Morgan Kaufmann Publishers, 1999: 7-10.

[18] LIU P, MIAO H. Theory of Test Modeling Based on Regular Expressions[M] // Structured Object-oriented Formal Lanuage and Method. 2013: 17-31.

[19] BELLI F, GROSSPIETSCH K E. Specification of fault-tolerant system issues by predicate/transition nets and regular expressions-approach and case study[J]. IEEE Transactions on Software Engineering, 1991, 17(6): 513-526.

(上接第 201 页)

结束语 本文基于 Intel Xeon Phi 协处理器集群平台实现了 GMRES 算法的移植工作。为克服加速设备内存不能满足计算要求的问题，本文首先对数据进行划分，采用分布式内存模型设计；为克服 GMRES 算法过多的集合通信带来的影响，对算法结构进行调整以便集合通信的隐藏。整个移植工作采用 MPI+OpenMP+offload 混合编程的模式：首先运用消息传递接口编程模型完成了数据任务的划分及并行设计，并利用计算与集合通信异步隐藏的方式提升并行算法的可扩展性，当进程数为 32 时，MPI 的并行效率达到 71.74%；接着运用 offload 编程模式，将计算任务卸载到 MIC 卡上，在 MIC 卡上通过开启 OpenMP 线程的方式实现并行；然后从 CPU 端与 MIC 端的数据传输、向量化及线程亲和性设计等方面对并行算法进行进一步优化；最后将并行算法应用到“局部径向基函数求解高维偏微分方程”的求解中。最终完成了对不同维度线性方程组的求解，测试结果表明，GMRES MIC 算法对于高维线性偏微分方程的求解具有收敛速度快、计算高效等特点。单块 MIC 卡的加速效果可达单颗 Xeon(R) E5-2650 v2 CPU 的 1.6 倍以上，4 块 MIC 卡的最高加速性能可达单颗 CPU 的 7 倍。因此，基于 MIC 异构集群平台的 GMRES 并行算法适用于超大规模线性方程组的求解，并有良好的并行加速效率和扩展性。

参 考 文 献

[1] SAAD Y, SCHULTZ M H. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems[J]. Siam Journal on Scientific & Statistical Computing, 1986, 7(3): 856-869.

[2] SAAD Y. Iterative methods for sparse linear systems(2nd ed)[M]. Philadelphia: SIAM, 2003: 151-126.

[3] NACHTIGAL N M, REICHEL L, TREFETHENK L NL N. A hybrid GMRES algorithm for nonsymmetric linear systems[J]. Siam Journal on Scientific & Statistical Computing, 1992, 13(3): 796-825.

[4] QUAN Z, XIANG S H. A GMRES based polynomal preconditioning algorithm[J]. Mathematical Numerical Sinica, 2006, 28(4): 365-376. (in Chinese).
全忠，向淑晃. 基于 GMRES 的多项式预处理广义极小残差法[J]. 计算数学，2006，28(4)：365-376.

[5] GHAEMIAN N, ABDOLLAHZADEH A, HEINEMANN Z. Accelerating the GMRES Iterative Linear Solver of an Oil Reservoir Simulator using the Multi-Processing Power of Compute Unified Device Architecture of Graphics Cards[C] // Proceedings of the 9th International Workshop on State of the Art in Scientific and Parallel Computing. Heidelberg: Springer, 2008: 156-159.

[6] WANG M L, KLIE H, PARASHAR M, et al. Solving Sparse Linear Systems on NVIDIA Tesla GPUs[C] // Computational Science-ICCS 2009 Lecture Notes in Computer Science. 2009, 5544: 864-873.

[7] LIU Y Q, YIN K X, WU E H. Fast GMRES-GPU Solver Large Scale Sparse Linear Systems[J]. Journal of Computer-Aided Design & Computer Graphics, 2011, 23(4): 553-560. (in Chinese)
柳有权，尹康学，吴恩华. 大规模稀疏线性方程组的 GMRES-GPU 快速求解算法[J]. 计算机辅助设计与图形学学报，2011，23(4)：553-560.

[8] GHYSELS P, ASHBY T J, MEERBERGEN K, et al. Hiding global communication latency in the GMRES algorithm on massively parallel computers[J]. Siam Journal on Scientific Computing, 2013, 35(1): 48-71.

[9] 王恩东，张清，等. MIC 高性能计算编程指南[M]. 北京：中国水利水电出版社，2012.

[10] JEFFERS J, REINDERS J. Intel Xeon Phi Coprocessor High Performance Programming[R]. morgan kaufmann. 2013.

[11] LI M, CHEN W, CHEN C S. The localized RBFs collocation methods for solving high dimensional PDEs[J]. Engineering Analysis with Boundary Elements, 2013, 37(10): 1300-1304.

[12] BELLALIJ M, REICHEL L, SADOK H. Some properties of range restricted GMRES methods[J]. Journal of Computational & Applied Mathematics, 2015, 290: 310-318.

[13] HE K, TAN X D, ZHAO H Y, et al. Parallel GMRES solver for fast analysis of large linear dynamic systems on GPU platforms [J]. Integration the VLSI Journal, 2016, 52(c): 10-22.