基于模型的若干逻辑边界覆盖测试准则

李丽萍[1] 李兴森[2]
(上海第二工业大学计算机与信息学院 上海 201209)[1] (浙江大学宁波理工学院管理学院 宁波 315100)[2]

摘 要 鉴于现有的基于规约的逻辑覆盖测试准则很少考虑到边界情况,对边界值分析法进行形式化,提出了一系列基于模型的逻辑边界覆盖测试准则。结果表明,相对于传统的逻辑覆盖测试准则,满足这些测试准则生成的测试用例能检测出系统更多的错误,既满足相应的逻辑覆盖测试准则,又能检测系统的边界情况。

关键词 软件测试,测试准则,逻辑表达式,边界测试,测试用例
中图法分类号 J **文献标识码** A

Model-based Logic Boundary Coverage Testing Criteria

LI Li-ping[1] LI Xing-sen[2]
(Computer and Information Institute, Shanghai Second Polytechnic University, Shanghai 201209, China)[1]
(Management School, Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China)[2]

Abstract Because the specification-based logic coverage criteria little regards to the boundary, this paper formalizesd the boundary value analysis method, proposed a series of model-based logical boundary coverage criteria. Results show test cases satisfying these criteria can detect more errors than original logic coverage testing criteria. They not only satisfy the logic coverage criteria but also test the system boundaries.

Keywords Software testing, Testing criteria, Logical expression, Boundary testing, Test case

1 引言

测试准则是一条或一组规则,这些规则定义了测试用例选择/生成的基本原则。测试准则的选择有基于程序的测试准则和基于规约的测试准则,或两者的结合。基于程序的测试准则针对程序的内部结构设计测试用例。基于规约的测试准则针对系统的形式规约设计测试用例。后者只检查系统功能是否按照规约书的规定正常运行,不考虑程序的代码。

逻辑表达式在各种软件产品中非常普遍,程序源代码、有限状态机、形式化的需求规约和测试规约中都有它的应用[1]。边界是特别容易导致系统出错的地方,边界值分析是一种有效而实用的方法,设计专门的测试用例,检测输入或输出域边界附近的情况,常常可以取得良好的测试效果。但是目前边界覆盖测试准则还不太成熟,基本没有形式化[2],它只是人们在进行功能测试(黑盒法)时采用的有启发性的测试方法,常与等价类划分方法结合使用,但很少用于自动化测试。

逻辑表达式是一种形式化的表示,常常用于自动化测试中。以前基于规约的逻辑覆盖测试准则很少考虑到边界情况。本文将逻辑覆盖测试准则和边界值分析结合起来,形式化边界值分析方法,提出了一系列基于模型的逻辑边界覆盖测试准则。目的是使基于这些准则生成的测试用例既能满足相应的逻辑覆盖又能检测到系统的边界情况。开发了一个相应的工具原型。结果表明,对比传统的逻辑覆盖测试准则,基于本文提出的测试准则生成的测试用例能检测出系统更多的错误。并且我们能根据不同的数值类型生成确定的测试数据。这些准则是对现有测试准则有益的补充和提高。

本文第2节给出了逻辑表达式的结构、基本的逻辑覆盖测试准则的定义,这些准则的定义摘自于文献[1],但为适应需要,描述方式稍有所改动;第3节介绍了边界和域的概念,并基于文献[2]讨论了基本的边界覆盖测试准则;第4节将逻辑覆盖测试准则和边界覆盖测试准则结合起来,提出了若干逻辑边界覆盖测试准则,并且给出了基于逻辑边界覆盖测试准则生成测试用例的方法;第5节讨论了相关的工作;最后得出结论并给出了进一步的研究方向。

2 基本逻辑覆盖测试准则

2.1 逻辑表达式

在基于形式规约的测试过程中逻辑覆盖测试准则是一组常用的测试准则。一般,逻辑表达式由包含了变量和关系运算符的谓词组成。变量主要是状态变量,关系运算符我们只考虑($=,\neq,<,>,\leqslant,\geqslant$)这6种。逻辑表达式的主要形式是谓词,谓词既可以是原子谓词也可以是复合谓词。

定义1(谓词,predicate) 一个谓词可以是布尔型变量、由关系运算符连接的非布尔型变量组成的关系表达式、返回布尔值的函数以及由逻辑连接词连接的它们的组合等项。

原子谓词是不包含任何逻辑连接词的布尔变量或表达式

本文受国家自然科学基金项目(71271191)资助。

李丽萍(1976-),女,博士,副教授,主要研究方向为软件测试、形式化方法,E-mail:liliping@sspu.edu.cn;李兴森(1969-),男,博士,教授,主要研究方向为数据挖掘、可拓学。

或其否定。复合谓词通过逻辑连接词¬(非)、∧(与)、∨(或)、⊕(异或)、→(蕴含)、↔(等价)将原子谓词相连。⊕(异或)、→(蕴含)、↔(等价)这3个逻辑连接词在源程序中比较少见,但是在规约语言中比较常见。一般可以将它们转换为等价于¬(非)、∧(与)、∨(或)的逻辑表达式。如:A→B等价于¬A∨B。

一个谓词由若干个文字组成,有些文献将不含任何逻辑连接词的谓词定义为子句(clause)。本文为了与经典一阶谓词逻辑的表示一致,采用了术语文字(literal)描述[3]。

定义2(文字,literal) 一个文字是不能再被分解为逻辑连接词连接的原子谓词。

文字是组成谓词的最小单元。例如,逻辑表达式$((X>Y)\wedge\neg A)\vee(X\leqslant Z)\vee F(x)$是一个谓词,其中,$(X>Y)$、$\neg A$、$X\leqslant Z$和$F(x)$是4个文字。

逻辑表达式是形式规约中用于描述前、后置条件的主要形式,也是程序中分支控制条件的主要表达形式,可以作为基于规约的测试用例研究的主要对象和定义测试充分性的基础[3]。假设P是一组谓词的集合,L是谓词P中所有文字构成的集合。对于P中的每个谓词$p\in P$,L_p是谓词p中所有文字的集合,这些文字按照在p中第一次出现的顺序构成,即$L_p=\{c|c\in L\}$,所以,$L=\bigcup_{p\in P}L_p$。

2.2 基本逻辑覆盖测试准则

测试覆盖准则是依据测试需求来定义的。测试需求是测试过程中测试用例必须满足或覆盖的一些确定的软件制品[4]。给定一系列测试需求TR,一个测试用例集TS(本文用TS表示测试用例集,主要是文字真值的组合)满足覆盖测试准则C当且仅当对TR中每个测试需求$tr\in TR$,存在至少一个测试用例$ts\in TS$满足tr。逻辑覆盖测试准则通过分析谓词和文字的真值关系来产生覆盖某些谓词和文字的真值取值的测试用例。基本的逻辑覆盖测试准则定义如下。

定义3(谓词覆盖,Predicate Coverage,PC) TS满足谓词覆盖准则,当且仅当对每个谓词$p\in P$,TS至少包含两个测试用例,其中一个使得p为true,另一个使得p为false。

对于谓词$p=((X>Y)\wedge\neg A)\vee(X\leqslant Z)\vee F(x)$,测试集$TS_1=\{(X>Y)=\text{true}\wedge A=\text{false})\wedge(X\leqslant Z)=\text{false}\wedge F(x)=\text{true},(X>Y)=\text{false}\wedge A=\text{false})\wedge(X\leqslant Z)=\text{false}\wedge F(x)=\text{false}\}$满足谓词覆盖。

定义4(文字覆盖,Literal Coverage,LC) TS满足文字覆盖准则,当且仅当对每个谓词$p\in P$,L_p中的文字$c\in L_p$,TS至少包含两个测试用例,其中一个使得c为true,另一个使得c为false。

对于谓词$p=((X>Y)\wedge\neg A)\vee(X\leqslant Z)\vee F(x)$,测试集$TS_2=\{(X>Y)=\text{false}\wedge(A=\text{true})\wedge(X\leqslant Z)=\text{true}\wedge F(x)=\text{true},(X>Y)=\text{true}\wedge A=\text{false})\wedge(X\leqslant Z)=\text{false}\wedge F(x)=\text{false}\}$满足文字覆盖。

谓词覆盖与文字覆盖都是很弱的覆盖测试准则,互不包含(subsume)。在以上的示例中,TS_1满足谓词覆盖但是不满足文字覆盖,TS_2满足文字覆盖但是不满足谓词覆盖。为了检查谓词中所有文字的真值组合,出现了文字组合覆盖准则。

定义5(文字组合覆盖,Literal Combinational Coverage,LCC) TS满足文字组合覆盖准则,当且仅当对每个谓词$p\in P$,TS包含L_p中所有可能的文字真值的组合。

例如,对于谓词$A\vee B$,测试集$\{A=\text{true}\wedge B=\text{true},A=\text{true}\wedge B=\text{false},A=\text{false}\wedge B=\text{true},A=\text{false}\wedge B=\text{false}\}$满足文字组合覆盖准则。文字组合覆盖准则包含谓词覆盖与文字覆盖。但该准则过于严格,很难在有限时间内完成测试用例的推导过程,也就是说,该准则不满足有限可应用性[3]。

有效谓词是决定系统行为的谓词,子域内的系统行为由有效谓词定义[5]。许多表达形式不同的形式规约都能被表达成有效谓词的形式。为了减少冗余谓词的数量,首先需要从形式规约中提取出有效谓词。我们可以将形式规约中的每个操作(迁移)划分为有效谓词形式,主要是前置、后置谓词形式。更精确地说,一个有效谓词包含两部分:定义子域的状态变量及其输入条件和在该子域中定义系统行为的谓词。

定义6(有效谓词,Effect Predicate) 一个有效谓词$EP_i=P\wedge Pre_i\wedge G_i\wedge Post_i$,$P$是不变式谓词,表示状态和输入变量$x_i$的取值范围,$Pre_i$和$Post_i$分别表示操作$i$的前置条件谓词和后置条件谓词,$G_i$表示操作$i$的守护条件谓词。

一个有效谓词对应于一个操作整个逻辑表达式的一个独立的析取项。覆盖一个形式模型中所有的有效谓词类似于覆盖程序中所有独立的路径。

3 基本边界覆盖测试准则

3.1 边界测试状态

不同的测试数据对诊断软件故障的能力有很大差别。为了提高测试效率,节约开发成本,减少测试用例的数量,必须从大量的输入数据中挑选出少数有代表性的测试数据,使得采用这些测试数据能够达到最佳的测试效果,高效地发现系统中隐藏的错误[7]。长期实践表明,边界是系统最容易出错的地方。一般,系统大量的错误不是发生在输入输出范围的内部,而是发生在输入或输出范围的边界上。所以,针对各种边界情况设计出高质量的测试用例可以测试出更多的错误,同时,也可以减少测试用例的数量。

一个谓词的边界值由它的域定义。目前,边界/域测试一般被用作测试生成算法的基础,在手工测试时经常使用,但是很少被形式化为覆盖准则。我们认为系统状态由n个状态变量$\{x_1,x_2,\cdots,x_n\}$的值唯一确定。对于输入变量x_i,每个变量都有其相应的取值范围D_i,所有输入变量取值范围的乘积构成了域D,即$D=D_1\times D_2\times\cdots\times D_n$。我们设定每个$D_i$都是有界的。因此,系统的可达状态空间是域$D$的子集。

定义7(边界状态) 假设$s=(x_1,x_2,\cdots,x_n)$是域D的一个状态,$D\subset\mathbb{R}^n$($\mathbb{R}^n$是实数域),s的邻居集合$N(s)=\{s,(x_1\pm\varepsilon,x_2,\cdots,x_n),(x_1,x_2\pm\varepsilon,\cdots,x_n),\cdots,(x_1,x_2,\cdots,x_n\pm\varepsilon)\}$,$\varepsilon$是一个大于0的极小值。我们说$s$是$D$的一个边界状态当且仅当$N(s)$中至少包含$\mathbb{R}^n\backslash D$的一个点,即如果域$D$中的某个状态的邻居已经不在域内,那么该状态就是域$D$的边界状态。

也可以说,一个状态s是域D的一个边界状态,如果对s的每个邻居$N(s)$,存在q',$q''\neq q$,使得$q'\in N(s)\cap D$,并且$q''\notin N(s)\cap D$。

定义8(域的边界,$Br(D)$) 域D的所有边界状态s的集合构成其边界$Br(D)$。更精确地说,$Br(D)$是边界状态s的集合,s的一些邻居落在域D中,而另一些邻居则落在域D外。

在定义 7 中，如果变量 $x_1, x_2, \cdots, x_n$ 取整数值，即 $x_i \in Z$，那么 $D \subset Z^n$ 是离散域。如果 s 是离散域的一个边界状态，那么定义 7 中 ϵ 的取值为 1。

域 D 的边界 $Br(D)$ 在简单例子中较容易确定，但是对于一个大型的系统，其边界状态的生成通常比较难。为了确定一个给定的点 $S \in D$ 是否是域 D 的一个边界状态，我们应该检测 $N(S) \backslash \{S\}$ 所有的点是否都属于 D。

文献[2]提出了一个基于代价最小的函数生成边界状态的方法。选择一个代价函数 f，并且在域 D 最小化（或最大化）f。因为最大化 f 等价于最小化函数 $(-f)$，我们在此只介绍最小化。对于一个合适的函数 f，最小化 f 使得找到一个边界状态成为可能，f 在 D 的最小值是 D 的一个边界状态。假设 $f(A)$ 是 f 在域 D 中的最小值，$A \in D$，即 $f(A) = \min\{f(Y) | Y(D}$，那么 A 就是域 D 的边界状态。

例如：代价函数为 $f(x_1, x_2) = x_1 + x_2$，对于整数域 $I = \mathbb{Z} \times \mathbb{Z}$，$n=2$，$P_1: -10 \leqslant x_1 \leqslant 10$，$P_2: 0 \leqslant x_2 \leqslant 20$，因为 $f(-10, 0) = \min\{f(x_1, x_2) | (x_1, x_2) \in D\}$，所以 $(-10, 0)$ 是域 D 的一个边界状态。

代价函数使得自动地生成边界状态成为可能。可以根据不同系统的实际情况构造不同的代价函数。

3.2 基本边界覆盖测试准则

从实际的角度看，覆盖准则对于何时停止测试提供了有用的规则。测试边界状态通常有 4 种方法：边界值分析、健壮性测试、最坏情况测试和健壮最坏情况测试[8]。这 4 种边界测试方法中前两种认为软件失效是由一个故障引发的，这样可能会遗漏一些常见的软件故障。后两种是基于多缺陷假设的测试，但是生成的测试用例数量太多，测试成本太高，缺乏可操作性。因为有些错误是基于单缺陷假设的测试所测试不出来的，所以本文考虑的是基于多缺陷假设的测试。本节首先基于文献[2]给出了形式化的边界覆盖测试准则，接着提出了基于模型的逻辑边界覆盖测试准则。

定义 9（所有边界覆盖，All Boundary Coverage，ABC） TS 满足域 D 所有边界覆盖准则，当且仅当对于域 D 中每个边界状态 $s \in Br(D)$，TS 含有测试用例使得 s 被测试，即 $Br(D) \subseteq TS$。

定义 10（单边界覆盖，One Boundary Coverage，OBC） TS 满足域 D 单边界覆盖准则，当且仅当对于域 D 中至少一个边界状态 $s \in Br(D)$，TS 含有测试用例使得 s 被测试。

定义 11（多维边界覆盖，Multi-Dimensional Boundary Coverage，MBDC） TS 满足域 D 多维边界覆盖准则，当且仅当对于域 D 中某些状态 $s \in D$，TS 含有测试用例使得 s 的每一个变量 $x_1, x_2, \cdots, x_n$ 分别取得其最大值和最小值。

如果域是有界的，那么必然存在域的边。边界状态一般存在域的边上。

定义 12（域的边） 域的边 $E_k = Br(D_k) \cap D$，D_k 是由约束 $P \wedge Pre_i$ 定义的域。

显然有 $Br(D) = \bigcup_{k=1}^{n} E_k$。

相应地，对应域的边有相应的边界覆盖准则。

定义 13（所有边覆盖，All Edges Coverage，AEC） TS 满足域 D 所有边覆盖准则，当且仅当对于 D 中每一条非空边 E_k，TS 含有测试用例使得该边上至少有一个边界状态被测试。

这等价于对于域 D 中每一条非空边 E_k，TS 满足其单边界覆盖准则。

定义 14（所有边多维覆盖，All Edges Multi-Dimensional Coverage，AEMD） TS 满足域 D 所有边多维覆盖准则，当且仅当对于 D 中每一条非空边 E_k，TS 含有测试用例使得每条边上至少有一个边界状态的所有变量 $x_1, x_2, \cdots, x_n$ 取得其最大值和最小值。

这等价于对于域 D 中每一条非空边 E_k，TS 满足其多维边界覆盖准则。

我们用一个简单的例子说明如何生成满足以上边界覆盖测试准则的测试用例。例如，一个离散域 D 定义为：$D = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} | (x^2 + y^2 \leqslant 16) \wedge (x + y \leqslant 4) \wedge (y \geqslant 0\}$。其测试用例设计如下：

$TS_1 = \{(4,0)\}$

$TS_2 = \{(4,0), (-4,0), (0,4)\}$

$TS_3 = \mathrm{Br}(D)$

其中，TS_1 只满足 OBC 准则，TS_2 满足除了 ABC 准则之外的其它准则，TS_3 满足以上定义的所有边界覆盖准则。

ABC 准则很强，对于一个大的系统很难满足，它意味着，首先要找到域 D 的所有边界状态，然后要获得并执行很多测试用例。OBC 准则是最弱的准则。

4 逻辑边界覆盖测试准则

4.1 逻辑边界覆盖测试准则

逻辑覆盖测试准则主要用于基于形式规约的测试。它通过分析规约中谓词和文字的真值关系来产生测试用例[4]。该形式规约由一系列状态和一系列改变状态的操作构成。边界状态是那些状态中至少有一个状态变量能在其子域中取到极值的状态。本文在基本逻辑覆盖测试准则的基础上，将逻辑覆盖测试准则与边界覆盖测试准则结合，提出了一系列逻辑边界覆盖测试准则。

定义 15（文字边界覆盖，Literal Boundary Coverage，LBC） TS 满足文字边界覆盖，当且仅当对每个谓词 $p \in P$，TS 中至少含有两个测试用例使得 L_p 中至少一个文字 c_i 的边界被测试并且使得 c_i 一次为 true，一次为 false。

例如，对于谓词 $p = (a \geqslant 3) \vee (b < 4)$，测试用例集 $\{a=3 \wedge b=3, a=2 \wedge b=3\}$ 满足文字边界覆盖测试准则。其满足了 $c_i = (a \geqslant 3)$ 的边界覆盖。

定义 16（谓词边界覆盖，Predicate Boundary Coverage，PBC） TS 满足谓词边界覆盖准则，当且仅当对每个谓词 $p \in P$，TS 中至少含有两个测试用例使得 L_p 中至少一个文字 c_i 的边界被测试并且使得 p 一次为 true，一次为 false。该文字 c_i 决定了谓词 p 的取值。

例如，对于谓词 $p = (a \geqslant 3) \vee (b < 4)$，测试用例集 $\{a=3 \wedge b=3, a=2 \wedge b=3\}$ 满足文字边界覆盖准则，但是不满足谓词边界覆盖测试准则。要满足谓词边界覆盖测试准则，首先需要找出能够使得谓词 p 取值为真和为假的文字真值的组合。测试用例集 $\{a=3 \wedge b=4, a=2 \wedge b=4\}$ 满足谓词边界覆盖，显然也满足文字边界覆盖准则。

文献[1]给出了决定的定义，对于谓词 p 中的一个文字 c_i，如果 p 中其余的文字 $c_j \in p, j \neq i$ 的真值使得改变 c_i 的真值会引起 p 的真值发生变化，那么称 c_i 决定谓词 p，其中 c_i

为主要文字，c_j 为次要文字。

利用文字和谓词之间的决定关系可以定义如下活动文字边界覆盖准则。

定义 17（活动文字边界覆盖，Active Literal Boundary Coverage，ALBC） TS 满足活动文字边界覆盖准则，当且仅当对每个谓词 $p\in P$ 和每个主要文字 $c_i\in L_p$，TS 中至少含有测试用例使得 L_p 中的每个主要文字 c_i 的边界被测试并且使得 c_i 一次为 true，一次为 false。

活动文字边界覆盖准则只要求主要文字的边界被测试，但是当主要文字的值为 true 和 false 时，次要文字的边界是否要求被测试，并且是否要求取相同的值？对于这一问题的不同回答产生了 3 个不同的逻辑边界覆盖测试准则。

定义 18（一般活动文字边界覆盖，General Active Literal Boundary Coverage，GALBC） TS 满足一般活动文字边界覆盖准则，当且仅当对每个谓词 $p\in P$ 和每个主要文字 $c_i\in C_p$，选取次要文字 $c_j\in C_p$，$i\neq j$，TS 中至少含有测试用例使得 L_p 中的每个主要文字 c_i 的边界被测试并且使得 c_i 一次为 true，一次为 false，而对于次要文字的边界不要求一定被测试到，并且其真值取值没有要求。

例如，对于谓词 $p=(a\geqslant 3)\vee(b<4)$，测试用例集$\{a=3\wedge b=3, a=2\wedge b=4\}$可以满足一般活动文字边界覆盖准则。

定义 19（相关活动文字边界覆盖，Correlated Active Literal Boundary Coverage，CALBC） TS 满足相关活动文字边界覆盖准则，当且仅当对每个谓词 $p\in P$ 和每个主要文字 $c_i\in L_p$，TS 中至少含有两个测试用例使得 L_p 中的每个主要文字 c_i 的边界被测试并且使得 c_i 一次为 true，一次为 false，并且所选的测试用例必须分别使得谓词 p 取不同的真值。

例如，对于谓词 $p=(a\geqslant 3)\vee(b<4)$，测试用例集$\{a=3\wedge b=3, a=2\wedge b=4, a=4\wedge b=3\}$可以满足相关活动文字边界覆盖准则。其中测试用例 $a=3\wedge b=3, a=2\wedge b=4$ 测试了 $a\geqslant 3$ 为主要文字的情况；测试用例 $a=2\wedge b=4, a=4\wedge b=3$ 测试了 $b<4$ 为主要文字的情况。

定义 20（受限活动文字边界覆盖，Restricted Active Literal Boundary Coverage，RALBC） TS 满足受限活动文字边界覆盖准则，当且仅当对每个谓词 $p\in P$ 和每个主要文字 $c_i\in L_p$，选择次要文字 c_j，$j\neq i$，以致 c_i 决定谓词 p 的值。对于 L_p 中的每个主要文字 c_i，TS 中至少含有两个测试用例使得 c_i 的边界被测试并且使得 c_i 一次为 true，一次为 false，并且这两个测试用例中次要文字必须取相同的值。

例如，对于谓词 $p=(a\geqslant 3)\vee(b<4)$，测试用例集$\{a=3\wedge b=4, a=2\wedge b=4, a=2\wedge b=3\}$可以满足受限活动文字边界覆盖准则。其中测试用例 $a=3\wedge b=4, a=2\wedge b=4$ 测试了 $a\geqslant 3$ 为主要文字的情况，次要文字 $b<4$ 都取了相同值 $b=4$；测试用例 $a=2\wedge b=4, a=2\wedge b=3$ 测试了 $b<4$ 为主要文字的情况，次要文字 $a\geqslant 3$ 取了相同值 $a=2$。

定义 21（文字边界组合覆盖，Literal Boundary Combinational Coverage，LBCC） TS 满足文字边界组合覆盖准则，当且仅当对每个谓词 $p\in P$，TS 中至少含有测试用例使得每个谓词中文字的边界的各种可能组合都至少出现一次。

文字边界组合覆盖是很严格的测试准则，对于安全攸关系统的测试可使用该准则。文献[9]指出在某型号飞机综合显示系统软件的第三方系统测试中，系统虽然进行了 100% 覆盖率的测试，各功能点及边界测试用例设计充分，并且利用因果图进行了功能组合测试用例的设计，但在测试结束后飞机试飞过程中仍然出现了一个软件边界值设计缺陷引起的故障。究其原因，主要是未考虑功能组合中可能会产生的边界条件的组合问题。也就是没有达到我们的文字边界组合覆盖测试准则。

以上提出的逻辑边界覆盖准则（定义 15－定义 21）中的边界既可以是文字的单边界也可以是多维边界。多维边界较单边界更为严格。

每个测试准则都有其优缺点。为了更详细地对测试准则进行比较，本文给出了以上逻辑边界覆盖测试准则之间的包含关系。假设 A 是所有测试准则的集合。令 $a_1, a_2\in A$，若一个测试集对于测试准则 a_1 是充分的就蕴含着它对于 a_2 是充分的，则称 a_1 包含 a_2，记作 $a_1\rightarrow a_2$。包含关系具有传递性。根据定义 15－定义 21 可知，文字边界组合覆盖（LBCC）是最严格的测试准则，包含其它的逻辑边界覆盖准则。文字单边界覆盖准则（LBC）是较弱的覆盖准则，只要求至少一个文字 c_i 的边界被测试到。谓词边界覆盖（PBC）要求文字 c_i 的边界被测试并且使得谓词 p 一次为 true，一次为 false，显然 c_i 是主要文字。与相关活动文字边界覆盖准则（CALC）类似，两者的区别是，谓词边界覆盖要求至少一个主要文字 c_i 的边界被测试，而后者每个主要文字 c_i 的边界被测试。显然满足相关活动文字边界覆盖准则的测试用例集一定满足谓词边界覆盖准则。从定义可以看出，满足受限活动文字边界覆盖准则（RALC）的测试用例集一定满足相关活动文字边界覆盖准则（CALC），而满足相关活动文字边界覆盖准则的测试用例集一定满足一般活动文字边界覆盖准则（GALC）。它们之间的包含关系如图 1 所示。

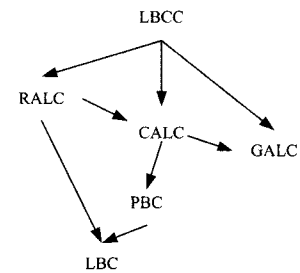

图 1　测试准则间的包含关系图

4.2　测试用例的生成

基于模型的系统从一个状态迁移到满足约束条件的另一个状态。我们首先把需求规约中的迁移（操作）表示成有效谓词的形式，有效谓词的 $P_i\wedge Pre_i$ 形成该操作的有效输入空间，作为测试的出发点，所有的测试数据必须来源于这个有效输入空间。这些描述系统状态子空间的一个有效谓词也称为边界目标[2]。接着在操作的有效输入空间中根据相应的覆盖测试准则设计测试用例。先要找出满足该准则的谓词和文字的真值组合，然后再通过求解约束的方式给出具体的测试数据[3]。

对于利用逻辑边界覆盖测试准则推导测试用例，我们同样先要找到满足该准则的谓词和文字的真值组合，然后再通过边界测试点的选择给出具体的测试数据。在边界测试点的选取方面，我们参考了 ON-OFF 边界测试点[2,6]的选取策略。ON 测试点是满足给定谓词边界条件的边界点；OFF 测试点

位于给定谓词边界之外，并且离 ON 点尽可能近。我们认为，ON 测试点是对输入变量取边界值使文字为 true，OFF 测试点是对输入变量取刚好超出边界的值(边界值±ε)使文字为 false，ε 是一个大于 0 的极小值。

测试一个不等式(关系运算符为<，>，≤，≥)谓词边界时，需要选取一个 ON 测试点和一个 OFF 测试点。测试相等或不等(关系运算符为=，≠)谓词边界时，需选取一个 ON 测试点和两个 OFF 测试点，两个 OFF 点分别位于被测边界的两侧，并且离 ON 点尽可能近。

对于简单的逻辑表达式，例如，对于谓词 $p=a\geqslant 3 \vee b<4 \wedge a\in \mathbb{Z} \wedge b\in \mathbb{Z}$，其边界测试点 ON-OFF 点的选取非常简单。对于文字 $a\geqslant 3$，其 ON 测试点是使文字取值为真的点 $a=3$，OFF 测试点是边界值为 -1，$a=2$；同理，对于 $b<4$，其 ON 测试点是 $b=3$，OFF 测试点是 $b=4$。

对于复杂的逻辑表达式，例如，$a_1x_1+\cdots+a_nx_n\leqslant b$，我们采用代价函数最小化的方法。设定代价函数 $f=a_1x_1+\cdots+a_nx_n$，然后找到使 f 取最小值或最大值的点，即边界点。

因为尽管任意值和边界值都能使有效谓词得到检测，但是众所周知边界是系统最“脆弱”的地方，应用边界值分析法能检测出系统中的更多错误。B. Legeard 等用工业界的实际例子 Smart Card GSM 11-11 standard[5]，Java Card transaction mechanism[10] 和 Metro/RER ticket validation algorithm[11] 等证明了对于同样数量的测试用例，使用边界测试能检测出系统更多的错误，大大节约测试成本。B. Legeard 等开发了一个自动生成边界测试用例的工具 BZ-TESTING-TOOLS(BZ-TT)，该工具已经应用于多个大型工业实例[5,10-11]的测试。

文献[12-15]提出了将基于边界的测试覆盖准则与基于数据流、控制流或者基于迁移的覆盖准则结合，基于 OCL 表达式自动生成边界测试用例，并且开发了相应的工具 ParTeG[15]。本文的思想与他们的工作非常相似，但并不是受其启发。在本文的修改期间，进一步查找文献时才发现了类似的工作。这也从侧面表明了本文研究工作的可行性。但是两者在研究细节上有区别。本文提出的这一系列基于模型的逻辑边界覆盖准则是运用域边界值测试逻辑表达式并保证其满足相应的逻辑覆盖测试准则，兼顾了两者的优点，生成的测试用例既测试了系统边界情况又满足了相应的逻辑覆盖准则。可适用于所有包含逻辑表达式的软件测试。S. Weißleder 等的工作主要针对 UML 的状态机图和 OCL 表达式，工作更加具体。边界值的选择只考虑输入参数值为 true 的情况，这样测试时可能会遗漏一些常见的软件故障。

为了便于比较，本文以经常用于测试举例的三角形[2,8,15]的例子来说明。系统要求用户输入 3 条边 a,b,c，根据输入边的长度不同生成不同的三角形。我们将每个行为表示成有效谓词的形式 $EP_i=P\wedge Pre_i\wedge G_i\wedge Post_i$。文献[15]用 UML 的状态机图描述了三角形的分类。在这个三角形的例子中，因为没有状态变量，所以没有不变式 P。$Pre=(1\leqslant a\leqslant MAXSIZE)\wedge(1\leqslant b\leqslant MAXSIZE)\wedge(1\leqslant c\leqslant MAXSIZE)$，MAXSIZE 是一个正整数常量，可根据实际情况指定。本文中 MAXSIZE=20。根据系统需求我们可以得到 8 个有效谓词[2]：

EP_1：Pre∧($a+b\leqslant c$)∧kind=不能组成三角形

EP_2：Pre∧($a+c\leqslant b$)∧kind=不能组成三角形

EP_3：Pre∧($b+c\leqslant a$)∧kind=不能组成三角形

EP_4：Pre∧($a+b>c$)∧($a+c>b$)∧($b+c>a$)∧($a=b$)∧($b=c$)∧kind=等边三角形

EP_5：Pre∧($a+b>c$)∧($a+c>b$)∧($b+c>a$)∧($a=b$)∧($b\neq c$)∧kind=等腰三角形

EP_6：Pre∧($a+b>c$)∧($a+c>b$)∧($b+c>a$)∧($b=c$)∧($a\neq b$)∧kind=等腰三角形

EP_7：Pre∧($a+b>c$)∧($a+c>b$)∧($b+c>a$)∧($a=c$)∧($b\neq c$)∧kind=等腰三角形

EP_8：Pre∧($a+b>c$)∧($a+c>b$)∧($b+c>a$)∧($a\neq b$)∧($b\neq c$)∧($a\neq c$)∧kind=一般三角形

在 EP_1 中，G_1 为($a+b\leqslant c$)，$Post_1$ 为 kind=不能组成三角形。EP_4 中，G_4 为($a+b>c$)∧($a+c>b$)∧($b+c>a$)∧($a=b$)∧($b=c$)，$Post_4$ 为 kind=等边三角形。

前置谓词是操作的前提，如果 8 个有效谓词中 Pre 均为 false，那么 Pre 后面的文字就被掩盖了，不能被测试到。所以在此我们先测前置谓词。a,b,c 边界值可以取上边界也可以取下边界，本文统一取下边界。根据 ON-OFF 策略设计测试用例，如表 1 所列。

表 1　前置谓词测试用例

编号	a	b	c	期望输出	实际输出
TS1	0	1	1	无效值	无效值
TS2	1	0	1	无效值	无效值
TS3	1	1	0	无效值	无效值

当测试 8 个有效谓词时我们设定每个有效谓词中的 Pre 均为 true。8 个有效谓词中的文字考虑了三角形 a,b,c 输入的所有情况。只要使每个文字在边界值处取到真值，就能保证满足文字边界覆盖。对应的测试用例如表 2 所列。测试用例 TS4－TS11 既满足文字边界覆盖又满足谓词边界覆盖。

表 2　有效谓词测试用例

编号	谓词	a	b	c	文字取值	谓词取值
TS4	EP1	1	1	2	T	T
TS5	EP2	1	2	1	T	T
TS6	EP3	2	1	1	T	T
TS7	EP4	1	1	1	T	T
TS8	EP5	2	2	1	T	T
TS9	EP6	1	2	2	T	T
TS10	EP7	2	1	2	T	T
TS11	EP8	3	4	5	T	T

在三角形的例子中，我们人为地植入了一些错误，通过比较发现本文提出的逻辑边界覆盖准则在检错能力上更优于传统的逻辑覆盖测试准则。

5　相关工作

边界测试通常与域测试联系在一起。近年来，国内关于边界测试自动化和边界覆盖准则形式化的相关工作还鲜有报道。国外主要有 B. Legeard 和 S. Weißleder 的工作。B. Legeard 等[2,4,5]提出了一系列面向数据的边界覆盖测试准则，开发了一个自动从 B，Z，UML/OCL 规约生成测试用例的工具 BZ-TT。该工具已在许多软件产品[10,11]中证明有效。本文的边界覆盖准则借鉴了他们的方法，但是我们将边界覆盖准则与逻辑覆盖准则结合在一起，提出了逻辑边界覆盖测试准则。

S. Weißleder 在他的博士论文[15]中详细讨论如何基于模

型针对 UML 状态机图、类图和 OCL 表达式自动生成测试用例。他提出了将基于边界的覆盖准则与基于数据流、控制流或者基于迁移的覆盖准则结合起来，并且开发了配套的工具 ParTeG[14]，生成的测试用例将满足基于数据流、控制流的覆盖准则和边界覆盖准则。本文的思想与之不谋而合。两者最主要的区别是他将状态图迁移上的守卫条件(OCL 表达式)转换为带输入参数的条件，然后使用这些条件，基于边界覆盖准则生成测试用例。文[12]提出了一种新的准则 MDCC，它将条件覆盖准则(CC)与多维边界覆盖准则(MD)结合。边界值的选择是针对输入参数值的范围采用面向分区的方法(Partition-Oriented testing)。本文提出的准则与他们的思想类似，但是本文主要针对逻辑表达式和边界覆盖准则，边界值的选择运用了基于边界域的 ON-OFF 方法，适用范围更广。

赵瑞莲[8]提出了一种基于谓词切片的字符串测试数据生成方法，即对选定路径上给定的字符串谓词，以相应的动态切片标准为准则，形成关于输入变量的谓词切片；对谓词中变量的每一个字符生成给定字符串谓词边界的 ON-OFF 测试点。本文与其不同，主要针对逻辑表达式，重点是逻辑边界覆盖测试准则的提出。

T-VEC Tester[16]是一个基于模型的自动化软件测试工具。T-VEC 也注重于系统的边界测试，倡导要覆盖软件的所有边界(We Cover All Boundaries)。这也说明了边界测试的自动化越来越引起人们的重视。

结束语 针对边界值分析很少应用于自动化测试的情况，本文将逻辑覆盖准则与边界值覆盖准则结合，提出了一系列基于模型的逻辑边界覆盖准则。基于这些准则可产生具体的包括测试数据和预期输出的测试用例。生成的测试用例既能满足相应的逻辑覆盖准则又能检测到系统的边界情况，并能控制从形式化模型中生成测试用例的数量。对应的工具原型已经实现。下一步的研究主要是对本文提出的测试准则进行度量与评估，并希望将测试工具原型集成到一些主流的测试工具中。

参 考 文 献

[1] Amman P, Offutt J. Coverage criteria for logical expressions[C]// Stephanie K, ed. Proc. of the 14th Int'l Symp. on Software Reliability Engineering. Washington: IEEE Computer Society Press, 2003: 99-107

[2] Kosmatov N, Legeard B, Peureux F, et al. Boundary coverage criteria for test generation from formal models[C]// Software Reliability Engineering, International Symposium. 2004: 139-150

[3] 刘玲，缪淮扣. 对逻辑覆盖软件测试准则的公理化评估[J]. 软件学报，2004，15(9)：1301-1310

[4] 钱忠胜，缪淮扣. 基于规约的若干逻辑覆盖测试准则[J]. 软件学报，2010，21(7)：1536-1549

[5] Bouquet F, Legeard B, Vacelet N, et al. Faster Analysis of Formal Specifications[C]// Proceedings of the 6-International Conference on Formal Engineering Methods (ICFEM'04), LNCS. Seattle, USA: Springer Verlag, November 2004

[6] Legeard B, Peureux F, Utting M. Automated Boundary Testing from Z and B[C]// Proceedings of the International Conference on Formal Methods Europe (FME'02), volume 2391 of LNCS. Copenhagen, Denmark: Springer Verlag, 2002: 21-40

[7] 赵瑞莲. 软件测试方法研究[D]. 北京：中国科学院，2001

[8] Jorgensen P C. 软件测试[M]. 韩柯，杜旭涛，译. 北京：机械工业出版社，2003：70-75

[9] 刘畅，王轶辰，刘斌，等. 软件边界组合测试的典型案例分析[J]. 计算机工程与应用，2009，45(20)：74-77

[10] Bouquet F, Legeard B. Reification of executable test scripts in formal specification-based test generation: the Java Card transaction mechanism case study[C]// Proceedings of the International Conference on Formal Methods Europe (FME'03), volume 2805 of LNCS. Pisa, Italy: Springer Verlag, 2003: 778-795

[11] Caritey N, Gaspari L, Legeard B, et al. Specification-based testing-Application on algorithms of Metro and RER tickets (confidential)[R]. Technical Report TR-03/01. LIFC-University of Franche-Comt'e and Schlumberger Besancçon, 2001

[12] Weißleder S, Sokenou D. Automatic Test Case Generation from UML Models and OCL Expressions[C]// Testing of Software-From Research To Practice (associated with Software Engineering 2008). February 2008

[13] Weißleder S, Schlingloff B-H. Quality of Automatically Generated Test Cases based on OCL Expressions[C]// ICST. April 2008

[14] Weißleder S. ParTeG (Partition Test Generator)[OL]. http://parteg. sourceforge. net. Oct 2009

[15] Weißleder S. Test models and coverage criteria for automatic model-based test generation with uml state machines[D]. Präsident der Humboldt-Universität zu Berlin. 2010

[16] T-Vec: We Cover All Boundaries [OL]. http://www. t-vec. com/

(上接第 90 页)

[7] Maji S. Efficient Classification for Additive Kernel SVMs[J]. Pattern Analysis and Machine Intelligence, 2013, 35(1)

[8] Erdmann M, Nguyen D D. Hierarchical Training of Multiple SVMs for Personalized Web Filtering[C]// PRICAI 2012: Trends in Artificial Intelligence. 2012

[9] Maldonado S, L'Huillier G. SVM-Based Feature Selection and Classification for Email Filtering[M]. Pattern Recognition-Applications and Methods, 2013

[10] 许高建. 基于 Web 的文本挖掘技术研究[J]. 计算机技术与发展，2007，17(6)

[11] 申红，吕宝粮. 文本分类的特征提取方法比较与改进[J]. 计算机仿真，2006，23(3)

[12] 闭乐鹏，徐伟，宋瀚涛. 基于一类 SVM 的贝叶斯分类算法[J]. 北京理工大学学报，2006，26(2)

[13] 刘文，吴陈. 一种新的中文文本分类算法-One ClassSVM—KNN 算法[J]. 计算机技术与发展，2012

[14] Yang Y, Pedersen J O. A comparative study on feature selection in text categorization[M]. Machine Learning-International, 1997

[15] Manevitz L M. One-class svms for document classification[J]. The Journal of Machine Learning Research, 2002, 2: 139-154

[16] Chang C-C, Lin C-J. LIBSVM: A library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3)

[17] Li Wen-kai. A Positive and Unlabeled Learning Algorithm for One-Class Classification of Remote-Sensing Data[J]. Geoscience and Remote Sensing, IEEE Transactions on, 2011, 40(2): 717-725