文章编号:1005-0523(2009)04-0082-05

面向对象类测试方法研究

赵丽萍,汤文亮

(华东交通大学 软件学院,江西 南昌 330013)

摘要:面向对象的软件开发给测试带来了新的挑战。类级测试是面向对象测试过程中的一个重要阶段。本文研究了基于代数规格说明的面向对象类测试方法,构造了一个半自动化的测试框架。公理系统是代数规格说明中的最重要部分,该框架主要基于CLA算法来完成代数规格说明中公理系统的设计和实现,采用GNF方案实现面向对象类级测试中范式半自动化辅助生成工具的设计和实现,最后在此基础上,将GFT算法实现为半自动化辅助工具。

关 键 词:面向对象测试;代数规格说明;测试用例;公理系统

中图分类号:TP311.5 **文献标识码**:A

面向对象软件是一种新的减少成本、提高可用性、灵活性和高效的软件开发方法。面向对象程序设计具有继承、数据抽象、动态联编和信息隐藏等特点[1]。因此面向过程软件测试和软件度量方法不适合面向对象软件测试和软件度量。由于形式规范具有严格、精确并支持机械推导的特点,将形式规范与测试相结合已成面向对象测试技术的重要发展方向,而代数规格说明是一种非常具有代表性的形式规范,所以本文主要对基于代数规格说明的面向对象类测试方法进行研究,并提出了自动化的测试框架。该框架不仅使范式的生成实现半自动化,还可以在构造范式的过程中随时剔除假范式,这在一定程度上降低了测试用例数,提高了测试效率。

1 代数规格说明

代数规格说明用于指明目标程序的功能。类的代数规格说明由两部分组成:语法声明部分和语义部分。前者列出包含的操作及相应的输入和输出参数的定义域;后者由等式公理(axiolns)组成,用来描述操作的行为特性[2]。

在代数规格说明中,一串合法操作序列称为项(term)。不含变量的项称为基项;一个不能再被规格说明中的任一公理重写的项称为范式(normalform)。每个项都有唯一范式的规格说明称为正则规格说明。

类 C 的创生子是带着 $i(i\geqslant 0)$ 个不同于类 C 的类 A_i 的对象作为参数,生成类 C 对象的操作。类 C 的构造子或转换子是带着 $k(k\geqslant 0)$ 个与 C 相同或不同的类 D_k 的对象作为参数,作用到类 C 对象上,得到的结果仍为类 C 对象的操作。若该操作能出现在范式中则称为构造子,否则称为转换子。类 C 的观察子是带着 $k(k\geqslant 0)$ 个与类 C 相同或不同的类 D_k 的对象作为参数,作用到类 C 对象上得到的结果为不同于类 C 的类 E 的对象的操作。对于一个正则规格说明,两个基项 u_1 和 u_2 称为等价(表示为 $u_1 \sim u_2$),当且仅当它们能通过某些作为重写规则的公理重写成相同的范式。

2 类级测试用例自动化生成的框架

类级测试用例自动化生成的框架如图1所示。

2.1 代数规格说明的公理产生算法(CLA)

采用CLA(Construction of Left Side of Axioms)算法来为面向对象类测试辅助产生代数规格说明公理系统的左边部分。算法CLA[3]:

收稿日期:2009-06-04

作者简介:赵丽萍(1981-),女,山西朔州人,硕士,助教,研究方向为软件工程。

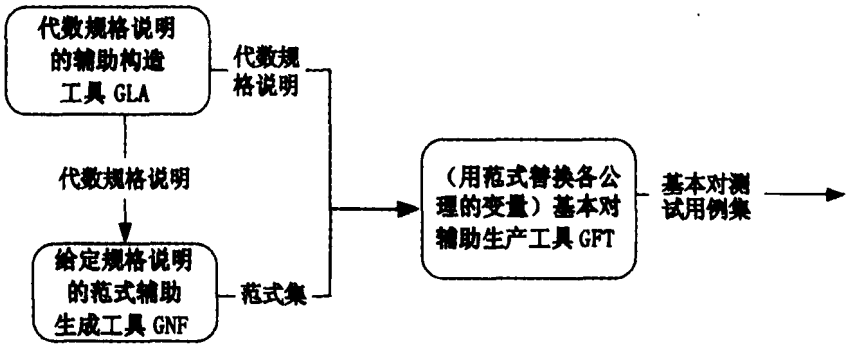

图1　类级测试用例生成的自动化框架

(1) 对于给定的一个类,通过和需求分析员的交互,确定并且输入产生子的集合 CR、观察子的集合 OB 以及变化子的集合 CT。

(2) 把公理的"前缀式"集合标记为 PL,所谓公理的"前缀式"是指一个集合,这个集合里面的元素是不能作为公理的左边,需要在其后面加一个操作后才有可能成为公理的左边。首先置集合 $PL=\varphi$,然后把公理系统的"预备公理集"标记为 PA。所谓的"预备公理集"是指一个集合,这个集合里面的每一条都是已经生成的"待定的"公理,预置 $PL=\varphi$。

(3) 对于 CR 里的每一个元素 $cr\in CR$,对于 OB 中的每一个元素 $ob\in OB$,做:

构造一条公理 ax,公理以 $cr.ob$ 为左边,由分析员判断得到与之相应的右边以及条件,如果无条件,则不用输入条件。将 ax 加入到集合 PA,$PA=PA\cup\{ax\}$。如果条件为非 true,之后则需要手动添加相反条件的对应公理。

(4) 对于 CR 里的每一个元素 $cr\in CR$,对于 CT 中的每一个元素 $ct\in CT$,做:

| 分析员判断 $cr.ct$ 是否能成为预备公理集中的公理左边;

如果可以,则

| 构造一条公理 ax,公理以 $cr.ct$ 为左边,由分析员给出与之相应的右边以及条件,如果无条件,则不用输入条件;

调用一致性和独立性检验算法,检查 ax 是否是和集合 PA 是一致的和独立的。

| 如果是,则将 ax 加入到集合 PA,$PA=PA\cup\{ax\}$;

如果条件为非 true,之后则需要手动添加相反条件的对应公理。如果 ax 的条件为非 true,把这条公理加入到 PL,$PL=PL\cup\{cr.ct\}$;

否则,分析员检查是否要重新给出 ax 的公理左边和条件,或者删除、修改现在已有的侯选集队 PA 中的与 ax 造成非一致或非独立的公理 ax_0,重新生成或者跳过这条公理,把这条公理加入到 PL,$PL=PL\cup\{cr.ct\}$。| |

如果不可以,则跳过这条公理,把这条公理加入到 PL,$PL=PL\cup\{cr.ct\}$。|

(5) 取一个本类的变量 A,且把这个变量加入到"前缀式"集,$PL=PL\cup\{A\}$,$PL_0=PL$。

(6) 对于每一个 $X\in PL$,对于每一个 $ct\in CT$,做

| $X=X.ct$;

分析员判断 X 是否能成为预备公理集中的公理左边;

如果可以,则

| 构造一条公理 ax,公理以 X 为左边,由分析员给出与之相应的右边以及条件,如果无条件,则不用输入条件。

调用一致性和独立性检验算法,检查 ax 是否和集合 PA 是一致的和独立的。

| 如果是,则将 ax 加入到集合 PA,$PA=PA\cup\{ax\}$;如果条件为非 true,则需要手动添加相反条件的

对应公理。此时如果 ax 的总条件为非 true,把这条公理加入到 PL,$PL = PL \cup \{X\}$。

否则,分析员检查是否要重新给出 ax 的公理左边条件,或者删除、修改现在已有的候选集 PA 中与 ax 造成非一致或非独立的公理 ax_0,重新生成。否则跳过,把这条公理加入到 PL,$PL = PL \cup \{X\}$。} }

如果不可以,则跳过,把这条公理加入到 PL,$PL = PL \cup \{X\}$。}

(7) 对于每一个 $X \in PL$,对于每一个 $ct \in CT$,做:

$X = X.ct$;对于每一个 $ob \in OB$,做:

{ 如果在 PA 中存在以 X 为左边的公理 ax,并且 ax 的条件为 true,则跳过这条公理。

分析员判断 $X.ob$ 是否能成为预备公理集中的公理左边;

如果可以,则

{ 构造一条公理 ax,公理以 $X.ob$ 为左边,由分析员给出与之相应的右边以及条件,如果无条件,则不用输入条件。

调用一致性和独立性检验算法,检查 ax 是否与集合 PA 是一致的和独立的。

{ 如果是,则将 ax 加入到集合 PA,$PA = PA \cup \{ax\}$。如果条件为非 true,则需要手动添加相反条件的对应公理。

否则,分析员检查是否要重新给出 ax 的公理左边和条件,或者删除、修改现在已有的候选集 PA 中的与 ax 造成非一致或非独立的公理 ax_0,重新生成,或者跳过这条公理。(不需把 $X.ob$ 放入 PL) } }

如果不可以,则跳过这条公理。}

(8) $PL = PL - PL_0$;$PL_0 = PL$;

如果 $PL = \varphi$,转到步骤(10);

否则转到步骤(9)。

(9) 分析员判断是否停止预备公理集的构造,如果停止,转到步骤(10);否则,转到步骤(6)。

(10) 通过和需求分析员的交互,如有需要,对预备公理集 PA 进行处理,包括选择、检查、合并、提炼、修改变量名、增加、删除等。最后得到的公理集合 PA 就是代数规格说明的公理部分。

(11) 结束算法。

2.2 GNF 方案

用于测试面向对象程序基于代数规格说明的范式的半自动化生成方案,GNF 方案(Generating Normal Forms)的主要步骤[4]:

(1) 分析类 C 给定的代数规格说明的语法部分,确定生成子的集合 CR、构造子和转换子(未加以区分)的集合 CT,并从代数规格说明的公理部分得到给定类 C 的公理集合 AX。这些集合中的操作也许包含参数。

(2) 假定 PNF 表示到目前为止生成的所有预备范式的集合,PNF_0 表示在"上一次执行步骤(3)"后的预备范式集,PNF_1 代表在"这一次执行步骤(3)中即将生成的预备范式集。初始化 $PNF = CR$,$PNF_0 = CR$,$PNF_1 = NULL$。其中,对每一个 $cr \in CR$ 中的参数都要实例化。

(3) 对每一个 $t \in PNF_0$,重复做:

{ 对每一个 $ct \in CT$(其中,ct 中若有参数也要实例化),重复做:

{ 检查项 $t.ct$ 是否与 AX 中的某条公理的左边相匹配;若匹配,则跳过这一项 $t.ct$;

否则,$PNF_1 = PNF_1 \cup \{t.ct\}$。 }

$PNF = PNF \cup PNF_1$;

$PNF_0 = PNF_1$;

$PNF_1 = NULL$;

(4) 与分析员交互,确定是否继续生成预备范式,如果是,转(3);否则,转(5)。

(5) 与需求分析员交互,检查是否需要对集合 PNF 中的预备范式集进行选择、检查、合并、精简、增加

或删除等操作,以得到给定类 C 所要求的范式集 RNF。

(6) 输出集合 RNF 中的给定类 C 所要求的范式。

(7) 结束。

根据 GNF 算法,设计的原型系统的流程图如图 2 所示。

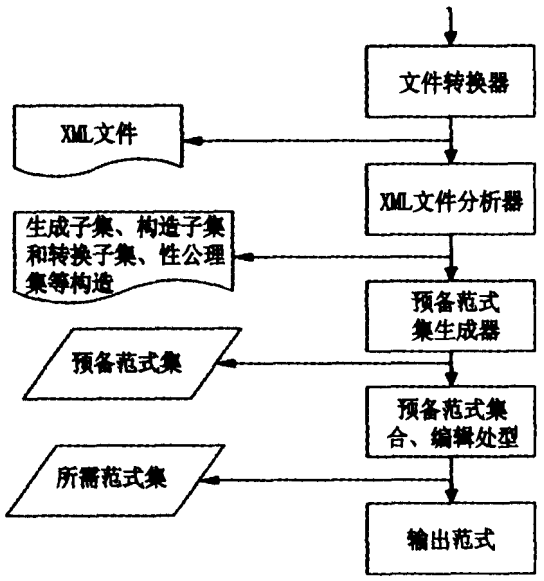

图2　GNF 方案流程图

2.3　GFT 算法

一般而言,代数规格说明的任一公理的前提条件并不总是为真,它可以是比较、逻辑等关系,形成分支条件。根据分支条件,可取无限个不同的范式代入公理相应的变量中,将会产生无限个的测试用例。利用包含 PDP(Path-based Domain Partition)的 GFT 算法可以较好地解决这个问题。通过 GFT 算法,可以生成有限数目的测试用例,从而有效降低测试用例生成的复杂度并且不失有效性。

假设给定类 T 的代数规格说明包含公理 $a_1, a_2, \cdots, a_n$,对每一个公理 $a_i(i=1,2,\cdots,n)$,GFT 算法(Generating a Finite Number of Test Cases)通过以下步骤生成作为测试用例集[5]:

(1) 若公理 a_i 中类 T 的变量 V 是不可观察的,则根据代数规格说明的语法部分,获得类 T 的创生子、变化子,利用 GNF 生成长度不超过 K 的所有范式的集合 NF。其中,K 的长度由需求分析员指定。

(2) 设 NF 包含范式 $nf_1, nf_2, \cdots, nf_m$,对每一个范式 $nf_j(j=1,2,\cdots,m)$,将 nf_j 取代公理 a_i 中的所有类 T 的变量 V,构成新的等式 a_{ij}。

(3) 若等式 a_{ij} 包含指定的操作 f,则根据操作 f 的定义将其输入划分为子域,确定每个变量的取值范围,并形成一组新的等式。

(4) 对每一个等式 a_{ij},将其所有变量在其取值范围内赋值,从而获得 a_{ij} 推导生成的不包含变量的一组测试用例。

(5) 若以上生成的一组测试用例可暴露错误,则退出本算法,否则执行(6)。

(6) 设(2)中的操作 f 的程序实现的方法为 m_f,将 PDP 作用于 m_f 以选择输入数据,并替代 a_{ij} 中对应的变量,从而获得公理 a_i 的另一组测试用例。

事实上,任何一种测试方法均不可能暴露程序中的所有错误,因而必须采用多种方法相结合的测试,才能尽可能的揭露程序中的错误,而步骤(5)及(6)便是对 GFT 方法中生成测试用例的一种补充方法。

根据 GFT 算法,设计的系统流程图如图 3 所示。

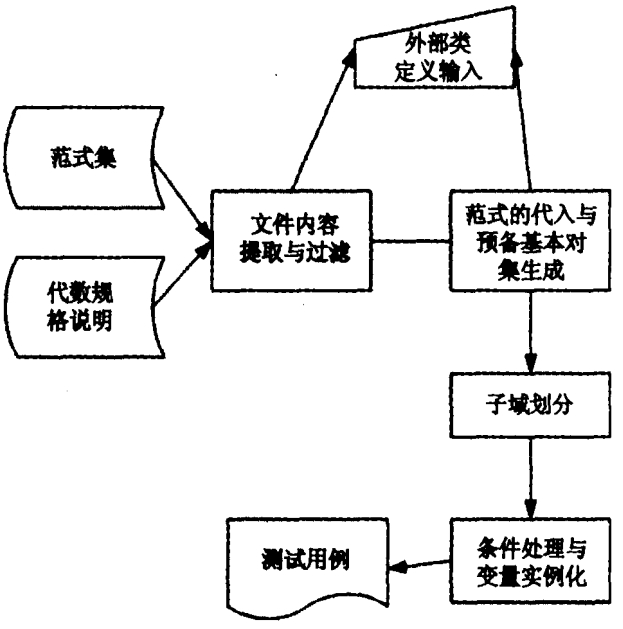

图3　GFT 算法流程图

2.4　测试实例

为了验证所提出的方法在实际测试工作中的可行性和实际测试效果,以电梯系统为例,从中简

单地说明测试用例的自动化生成，见图4。

以 *Door* 类为例，形式化描述为：

schema Door =

class

type Doors

value

$open: T.floor * Doors \rightarrow Doors$,

$close: T.floor * Doors \rightarrow Doors$,

$door\text{-}state: door \rightarrow T.floor \rightarrow T.Door\text{-}state$

axiom

[*door-state-open*]

$\forall f_1, f_2: T.floor, s.door.$

$door\text{-}state(close(f_1, s))(f_2) \equiv$

$if\ f_1 = f_2\ then\ T.open\ else\ door\text{-}state(s)(f_2)$

end

[*door-state-close*]

$\forall f_1, f_2: T.floor, s.door.$

$door\text{-}state(close(f_1, s))(f_2) \equiv$

$if\ f_1 = f_2\ then\ T.shut\ else\ door\text{-}state(s)(f_2)$ *end*

end

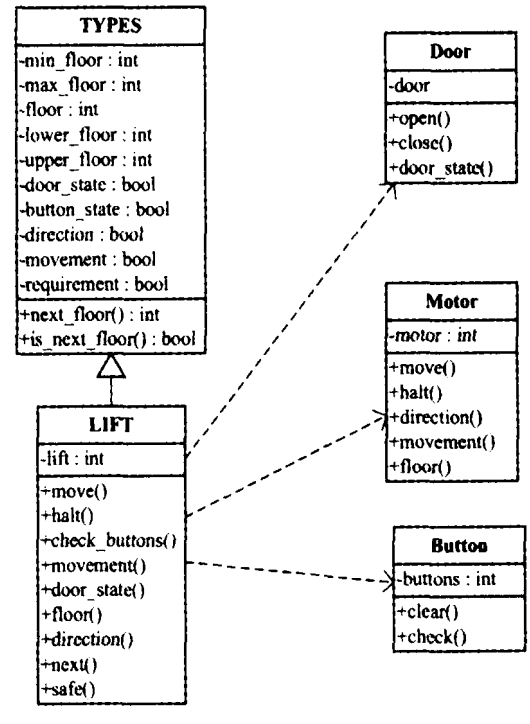

图4 电梯类图

对于楼门的描述中，只有两个动作：*open* 和 *close*，以及状态变量 *door-state*。对应的描述中给出了两个相关性条件：*door-state-open* 和 *door-state-close*。

根据测试用例的生成规则可以对应的产生四个测试用例：

(1) $f_1 = f_2 \rightarrow door\text{-}state(open(f_1, s))(f_2) \equiv T.open$

(2) $f_1 \neq f_2 \rightarrow door\text{-}state(open(f_1, s))(f_2) \equiv door\text{-}state(s)(f_2)$

(3) $f_1 = f_2 \rightarrow door\text{-}state(close(f_1, s))(f_2) \equiv T.shut$

(4) $f1 \neq f2 \rightarrow door\text{-}state(close(f_1, s))(f_2) \equiv door\text{-}state(s)(f_2)$

这两个用例保证了在楼门的 *open* 和 *close* 操作必须遵守这样的规则：在同一时刻只有一个楼门可以打开，而且打开的楼门必须是电梯所在楼层的楼门。

对于一个电梯系统而言，这个规则是必须遵守的，所以测试者可以在任何楼门的 *open* 和 *close* 操作执行时使用这两个测试用例对系统进行相关性条件检查，以保证系统的安全运行。

3 结语

主要研究了使用代数规格说明的形式化软件测试方法。采用 CLA 方案产生代数规格说明，采用 GNF 方案使范式的生成半自动化，最后利用所产生的代数规格说明和范式，利用 GFT 算法生成测试用例，实现了测试用例生成的半自动化生成。该框架能够使规格说明产生测试用例过程的自动化程度得到提高，有利于改善软件测试的效率，降低软件成本。

参考文献：

[1] 赵 良，叶俊民，罗 景，陈 利.一种面向对象类级状态测试的形式化方法研究[J].海军工程大学学报，2004，16(5)：77－81.

[2] 胡 煌，李远杰，曾 明，朱 利.基于公理系统的面向对象自动测试研究[J].微电子学与计算机，2005，22(7)：168－170.

(下转第99页)

参考文献:

[1] Ohshima T, Iwasaki T, Maegawa Y, Yoshiyama A & Mashima K. Transesterification of various methyl esters under mild conditions catalyzed by tetranuclear zinc cluster[J]. J Am Chem Soc, 2008, 130(10): 2 944 - 2 945.

[2] Gorka P & ScottJ Miller. Triumph of a chemical underdog[J]. Nature, 2008, 452: 415 - 416.

[3] 陈景文,王帅杰,陈　硕.应用PM3算法对部分有机锡化物的QSPR和QSAR研究[J].大连理工大学学报,2004,40(3):124 - 127.

[4] 余训爽,周享春,余训明.PM3原子电荷与氯代苯的理化生物活(毒)性的相关性研究[J].长江大学学报,2006,3(2):38 - 41.

Selectivity Reaction of Amino and Hydroxyl of Cyclohexane — PM3 Method

XU Wen-yuan, RUAN Chun-xiao, HAO Wei

(School of Basic Sciences, East China Jiaotong University, Nanchang 330013, China)

Abstract: Semi-empirical PM3 calculation is used to study the charge distribution of each atom in the reaction system as well as the thermodynamic functions such as gibbs free energy and equilibrium constant calculations. The different selectivity (with or without catalyst) of the substitutes (amino and hydroxyl) on Cyclohexane is explained. The results show that the reaction can proceed in a wide temperature range, and the content of the products is 50%.

Key words: Cyclohexanol; Cyclohexylamine; zinc catalyst; PM3

(责任编辑:刘棉玲)

(上接第86页)

[3] 何美玲.用于面向对象类级测试的范式生成半自动化工具的研究[D].广州:暨南大学,2008.

[4] HuoYanChen, T H Tse, T Y Chen, TACCLE. A methodology for object-oriented software testing at the class and cluster levels[J]. ACM Transactions on Software Engineering and Methodology, 2001, 10(4): 56 - 109.

[5] H Y Chen. A scheme to aid generation of normal forms for testing object-oriented programs[R]. UK: Technical Report CSD-JNU. 2008.2.

Research on Object-oriented Class Testing Method

ZHAO Li-ping, TANG Wen-liang

(School of Software Engineering, East China Jiaotong University, Nanchang 330013, China)

Abstract: Object-oriented software development raises new challenges for testing. The class-level testing is an important aspect of the test. The paper describes the methods of the class-level testing for object-oriented software based on algebraic specification, and constructs a frame for semi-automatic testing frame. Axiom system is the most important part of algebraic specification. The frame intends to design and implement a tool to aid constructing algebra specification, based on the scheme CLA. With the GNF schema, the frame could semi-automatically generate normal forms for the class-level testing. Finally, the algorithm GFT becomes a semi-automatic aided tool.

Key words: object-oriented testing; algebraic specification; test cases; axiom system

(责任编辑:刘棉玲)