软件测试 Software Testing

11. 路径测试

程适

cheng@snnu.edu.cn

计算机科学学院

2016 年 10 月 15 日

陕西师範大學
SHAANXI NORMAL UNIVERSITY

Outline

- 程序结构分析
- DD-路径
- 测试覆盖指标
- 基路径测试

结构性测试

- 逻辑覆盖
- 路径测试
- 数据流测试

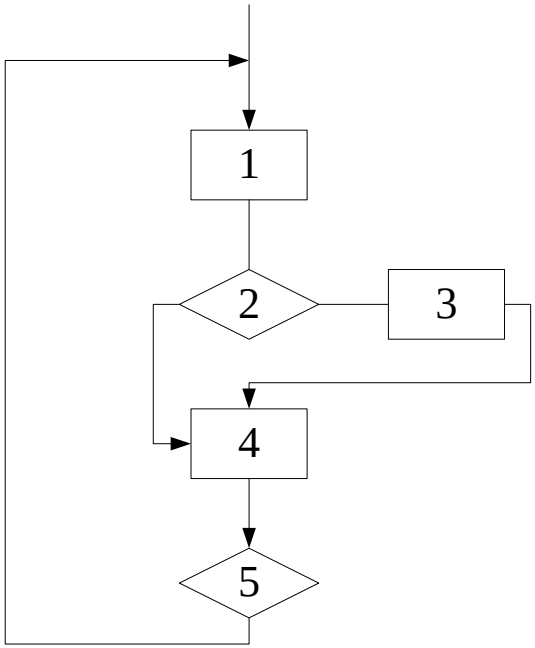
程序结构分析

- 控制流分析
  - 由于非结构化程序会给测试带来许多不必要的困难，所以业界要求写出的程序具有良好的结构。
  - 上个世纪70年代以来，结构化程序的概念逐渐被人们普遍接受。体现这一要求对某些语言并不困难，比如Pascal、C、Java，因为它们都具有反映基本控制结构的相应得控制语句。
  - 但对于有些开发语言要做到这一点，程序人员就要很注意程序结构化的要求，比如说汇编语言，若使用汇编语言编写程序，开发人员就尤其要注意程序的结构化要求。

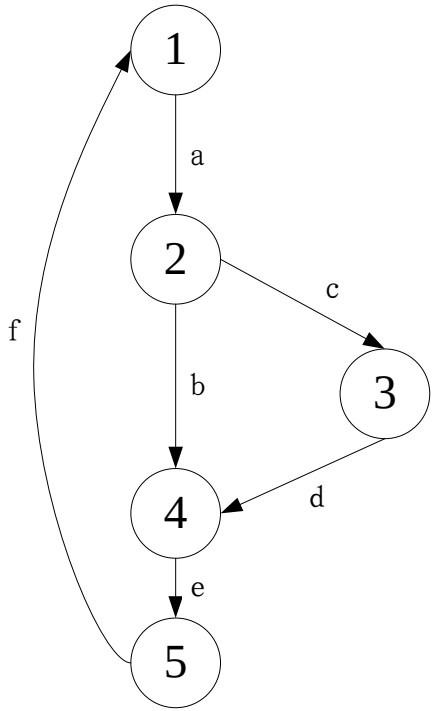
程序结构分析

- 程序流程图: 程序流程图又称框图，是一种程序控制结构的图形表示。在这种图上的框里面常常标明了处理要求或者条件，但是，这些标注在做路径分析时是不重要的。
- 控制流图: 为了了更加突出控制流的结构，需要对程序流程图做一些简化。其中图（a）图示一个含有两个出口判断和循环的程序流程图，把它简化成（b）的形式，称这种简化了的程序流程图叫做控制流图。

程序结构分析



(a) 程序流程图



(b) 控制流图

程序结构分析

- 在控制流图中只有两种图形符号，它们是：
  1. 节点：以标有编号的圆圈表示。它代表了程序流程图中矩形框表示的处理、菱形表示的两个到多个出口判断以及两条到多条流线相交的汇合点。
  2. 控制流线或弧：以箭头表示。它与程序流程图中的流线是一致的，表明了控制的顺序。为了方便讨论，控制流线通常标有名字，如图中所标的a、b、c等。

程序图

- 定义:
  1. 给定一个采用命令式程序设计语言编写的程序，其程序图是一种有向图，其中：节点是程序语句，边表示控制流。
  2. 从节点i 到节点j 有一条边，当且仅当对应节点j 的语句可以立即在节点i 对应的语句之后执行。
- 程序图不是唯一的

程序结构分析

- 程序图的意义：
  - 程序的执行对应于从源节点到汇节点的路径。
  - 测试用例和测试用例所执行的程序部分之间有明确的对应关系。
- 程序图上的完备测试

路径测试

- 路径测试
  1. DD-路径测试
  2. 基路径测试
- 路径测试方法的突出特点，是它们都基于被测程序的源代码，而不是基于定义。
- 由于这种绝对化的基础，结构性测试方法支持严格定义、数据分析和精确度量。

DD-路径测试

- 决策到决策的路径（DD-路径）是指语句的一个序列，从决策语句的“出路”开始，到下一个决策语句的“入路”结束。
- 在这种序列中没有内部分支，因此对应的节点像排列起来的一行多米诺骨牌，当第一块牌推倒后，序列中的其他牌也会倒下。

DD-路径测试

- 链路径
  - 链是一条起始节点和终止节点不同的路径，并且每个节点都满足内度＝ 1 、外度＝ 1 。
  - 初始节点与链中的所有其他节点有 2 –连接，不会存在 1 –连接或 3 –连接。
  - 有一种长度为 0 的退化链情况，即链有一个节点和 0 条边组成。

DD-路径测试

- DD-路径的定义: DD-路径是程序图中的一条链，使得：
  - 情况1：由一个节点组成，内度=0。
  - 情况2：由一个节点组成，外度=0。
  - 情况3：由一个节点组成，内度≥2或外度≥2。
  - 情况4：由一个节点组成，内度=1并且外度=1。
  - 情况5：长度≥1的最大链。

DD-路径测试

- 对于给定的程序，可以使用多种不同的程序图，所有这些程序图都可以简化为惟一的DD–路径。

基于指标的测试

- 语句与判断测试: 程序图的所有节点都至少走过一次
- DD-路径测试: 每条DD-路径都被遍历(C1指标), 较长的DD-路径一般代表复杂计算，可结合功能性测试方法
- DD-路径的依赖对偶: DD-路径对偶的最常见的依赖关系是定义/引用关系，变量在一个DD-路径中定义，在另一个DD-路径中引用。
- 多条件覆盖
- 循环覆盖

简单循环

- 其循环的最大次数为n
- 测试：
  - 跳过整个循环
  - 只循环一次
  - 循环两次
  - 循环m次其中$m < n$
  - 分别循环$n-1$次，n次，$n+1$次

串接循环

- 如两个串接循环是独立的，则可分别采用简单循环的测试方法测试
- 不独立第一个循环计数器是第二个循环的初值，则可采用嵌套循环方法进行测试

嵌套循环

- 如果嵌套循环也采用简单循环的办法此时会随嵌套层数成几何级数增加导致不可测
- Beizer提出一种减少测试级数的方法
  - 从最内层循环开始测试，内层循环按简单循环策略,所有外层循环次数设到最小数
  - 由内向外一次向上回退一次嵌套循环,回退后进行测试，本层循环的所有外层循环仍取最小值，本层循环嵌套的循环取一次“典型”值
  - 继续向外回退，直到所有循环测试完毕

测试覆盖指标

- 循环测试：每个循环都包含一个判断，并且需要测试判断的两个分支：进入循环、退出循环
- 将边界值测试用于循环测试过程中
- 一旦测试了循环，就可将其压缩成一个单独的节点

基路径测试

- “基”的概念
  - 向量空间的基是相互独立的一组向量，基“覆盖”整个向量空间，使得该空间中的任何其他向量都可以用基向量来表示;
  - 一个向量空间，存在一个线性无关的向量组$\{x_1, \ldots, x_n, \ldots\}$，使得对所有空间中的向量，都能被这个组线性表示;
  - 因此，一组基向量在一定程度上可表示整个向量空间的本质：空间中的一切都可以用基表示，并且如果一个基元素被删除了，则这种覆盖特性也会丢失。

基路径测试

- “基”对测试的潜在意义：
  - 如果可以把程序看做是一种向量空间，则这种空间的基就是要测试的非常有意义的元素集合。如果基没有问题，则可以希望能够用基表达的一切都是没有问题的。

基路径测试原则

- 具有高圈复杂度的程序需要更充分的测试
- 最大可接受圈复杂度指标：$V(G) = 10$
- 如果被测试单元具有更高的复杂度：
  - 简化单元
  - 计划更充分的测试

基路径测试原则

- 改进程序设计方面要比改进测试方面有更大的作用
- 如果程序具有很好的结构性，则总是可以压缩为只有一条路径的图

路径测试指导方针

- 基路径测试给出了必须进行的测试的下限。
- 如果发现同一条程序路径被多个功能性测试性用例遍历，就可以怀疑这种冗余不会发现新的缺陷。
- 如果没有达到一定的DD-路径覆盖，则可以知道在功能性测试用例中存在漏洞。
- 利用源代码的性质标识合适的覆盖指标，然后再使用这些指标交叉检查功能性测试用例。
  - 对于具有复杂逻辑的模块可能选择多条件覆盖
  - 对于大量迭代处理的模块采用循环覆盖指标

小结

- 路径测试
  - 程序结构分析
  - DD-路径
  - 测试覆盖指标
  - 基路径测试

致谢

谢谢，欢迎提问!