一种基于WPDAG的回归测试选择方法

刘 芳，姚跃明

（保险职业学院，湖南 长沙 410114）

摘 要：提出了一种基于WPDAG的回归测试选择方法，利用压缩的全路径动态执行过程生成修改影响集，可以有效地提高测试选择精度，降低测试选择成本。

关键词：WPDAG；回归测试；测试选择；修改影响

中图分类号：TP311 **文献标识码**：A **文章编号**：1673-9833(2007)05-0029-04

Regression Test Selection Method Based on WPDAG

Liu Fang，Yao Yueming

(Insurance Vocational College，Changsha 410114，China)

Abstract：A regression test selection method based on WPDAG is proposed, it can increases the test selection precision and reduces the cost of testing effectively by changing impact set through the compressed whole-dynamic execution process.

Key words：WPDAG；Regression testing；Test selection；Change impact

软件测试是保证软件质量、提高软件可靠性的关键[1]，在软件开发过程中占重要地位。而回归测试是软件测试中的一个重要环节，其目的是保证软件被修改后能正常运行，且不会给软件质量带来不利影响[2]。通常使用的回归策略中，最简单的就是重测所有测试用例，但这样做的测试成本过于昂贵。

为了减少回归测试的代价，应该只需要对程序修改影响部分进行重测，即从原来的测试用例集中选取被影响的测试用例来进行测试。我们利用修改影响分析来鉴别出程序修改影响的部分。修改影响分析能改善对资源预算的精确性，减少软件维护的成本。在回归测试中，利用修改影响分析技术能找出程序修改影响的部分，从而选择被影响的测试用例进行重测，可减少测试工作量[3]。传统的基于依赖性的影响分析技术有基于调用图的分析、静态程序切片以及动态程序切片3种[4]。

作为基本技术的程序调用图用于预测程序修改所带来的影响时，代价相对而言比较低廉，但是它可能产生高度的不精确性，从而导致鉴别出不存在的影响以及漏掉本来存在的影响。静态切片比起调用图要精确得多，但是代价昂贵，而且如果目的是去预测出现在测试用例中的动态行为的影响，它可能返回过大的影响集。动态切片能够对具体的程序执行或操作片断预测其影响，这对于许多维护任务是很有用的，但是它牺牲了作为结果的影响评估的安全性。鉴于以上分析，这3种方法各有弊端，因此，提出一种新的性价折衷的，能够对预测影响分析提供帮助的方法是有一定的研究意义的。

1 基于依赖的效费比折衷的影响分析技术

调用图通常用来评估软件改变的潜在影响[4]。可设想在程序P的某个过程p上的修改，将对程序P的调用图G上从p可达的任意一个节点产生潜在的修改影响。G的传递闭包包括了所有在程序P过程中的潜在影响关系。调用图容易理解，创建也相对容易，且

收稿日期：2007-08-22

作者简介：刘 芳（1975-），女，湖南双峰人，保险职业学院讲师，主要从事软件可行性方面的研究.

调用图中的传递闭包也容易执行。然而，真正的调用行为要比调用图显示的复杂得多。调用行为包括：1）进入与退出一个过程而没用调用任何其它过程；2）进入一个过程，调用一个其它的过程，然后退出；3）进入一个过程，以不定的顺序调用了多个其它过程。一个调用图捕获的是被一个单独的过程调用的潜在结构，但是它忽略了调用的其它方面。因此，基于调用图的分析可能导致不精确的影响集。例如，如图1所示的调用图（从节点A到节点B的箭头说明过程A调用了过程B），过程A传递调用了所有其它的过程。我们不能够从这个调用图判断出什么情况下将导致修改的影响从A传播到其它的过程。

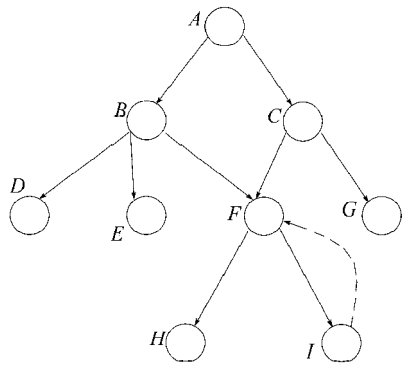

图1 调用图示例

Fig. 1 Example call graph

调用图的另一局限是由于过程的返回而不能捕获影响的传播。设想我们准备修改H，那么修改的影响可通过返回传播给F、C、B和A。一个典型的调用图没有返回的相关信息，仅靠推断得出信息将会导致错误。我们可推断G返回到F，但不能推断随着这个返回，影响是否同时传播给B和C，或它们中的一个。

与基于调用图的技术相比，基于静态切片的技术依赖于对细粒度依赖关系的考虑。静态切片是保守的、安全的，它考虑了所有可能的程序输入和行为。当要求安全性时，这种保守是重要的；但是，信靠这种结果的影响分析将会产生一个对维护人员没有多大用处的庞大的影响集。更为严重的是，所得到的影响集里还会有系统的预期操作中不会发生的影响的存在。对于一个不是以安全性来评价的系统来说，考虑与预期的用法相关的影响才是合算的。动态切片克服了静态切片的这些缺点。但是不论是动态切片还是静态切片，都要求细粒度的依赖分析，这都增加了额外的计算支持和设备的代价。此外，依赖的细粒度分析在许多预言性的影响分析情况下并不恰当。预言性的影响分析要求维护人员详细说明修改的近似定位，而知道修改部分的精确状态以及修改中涉及的精确变量是没有实际意义的。当修改涉及到删除代码时，在细粒度的级别上调用什么样的切片标准去预测潜在的影响很困难。在粗粒度级别的函数或者方法上工作，可简化影响分析任务，因为它只要求维护人员确定将会被修改的函数或方法的集合，而非要求修改精确句法。这就意味着调用级上的分析在实际应用中更加恰当。

总之，基于调用图和基于切片的方法都依赖于对源代码的访问来确定代码中的静态调用结构或依赖。在实际应用中，系统通常包含在源代码中不可用到的组件。

2 基于全路径的动态影响分析

使用在过程调用级别上收集动态路径信息将有助于定位上述问题。我们的方法可概述如下：如果我们准备修改过程p，我们就只会关注可沿着任意（并且仅仅）经过过程p的动态路径传递的影响。因此，任何在p之后调用或在p返回之后存在于调用堆栈之中的过程，均包括在潜在被影响的过程集中。

举例来说，假设我们有一个如图2字符串所示的单一执行轨迹，它的程序调用图如图1所示。在此字符串中，我们用r表示函数的返回，用x表示程序的退出。如果我们准备修改过程H，我们可以通过在轨迹中向前（或向右）检索直接或间接被H调用的过程以及向后（或向左）检索E所返回的过程来创建与此次执行相关的修改的动态影响。在下面这个例子中，过程H没有调用其它过程，但是H返回到F、B和A。H没有返回到I是因为I后面紧跟着一个r，I被调用后马上返回了。同样情况的还有G和C。因此，我们可以得出缘于过程H的修改而导致的潜在影响过程集合是{A，B，F，H}。

A	C	G	r	r	B	F	I	r	H	r	r	r	r	x

图2 单一执行轨迹

Fig. 2 Single execution trace

图2中给出的轨迹所代表的程序行为可以通过一个小的簿记记录下来。在轨迹中未紧接着相匹配的函数返回的过程意味着这个过程有调用。例如，从H向前统计的返回(r)比过程名称（包括H）多出来3个，这意味着有3个函数被返回。我们对这3个返回进行配对——向后检索没有紧邻返回的过程。比如，从H向后搜索查找3个过程名称，其间需要跳过C、G、I——因为它们的右方都遇到了紧邻的返回，这意味着在H被调用之前它们已经返回了。一个可供选择的方法可以鉴别出每一个过程所对应的返回。在这个例子中，我们可以通过在轨迹中向前检索对影响进行评估。

复合执行轨迹可以看作是多个单一执行轨迹的连接，其中每一子段的处理和单一执行轨迹没有什么不

同，只不过向前或者向后检索的时候不能到达两头的端点。举例来说，图 3 给出了一个系列的执行轨迹（可以看作是 3 个单一执行轨迹的连接）。如果我们修改过程 D，那么潜在受影响的过程就是 D、B、A。如果我们准备修改过程 F，那么从第一个轨迹得到的潜在受影响的过程集是 $\{A、B、F、I、H\}$，从第 3 个轨迹得到的潜在受影响的过程集是 $\{A、C、F、H\}$，最终的合集就是 $\{A、B、C、F、I、H\}$。

这种方法存在的一个明显问题是执行的路径将会非常长。因此，实际技术中就必须保证轨迹的长度是可行的。在下一部分，我们将通过构建一个称之为全路径 DAG 的直接非循环图来进行压缩的方法，大幅度减少轨迹的长度，然后产生一个算法，直接通过此 DAG 图计算影响分析。

A	C	G	r	r	B	F	I	r	H	r	r	r	r	x	A	B	D	r	r	r	x	A	C	F	H	r	r	r	r	x

图 3　复合执行轨迹

Fig. 3　Multiple execution traces

3　算法

我们要求程序产生一个复合轨迹，其中依序包含过程名称、函数返回及程序的退出。SEQUITUR 数据压缩算法[5]产生一个语法规则，用来大幅度减少轨迹的规模。由 Larus[6]发展的全路径 DAG 图表示法被用来产生这个语法规则。

3.1　SEQUITUR 数据压缩算法和全路径 DAG 图

SEQUITUR 数据压缩算法检查由程序产生的轨迹，然后通过建立一个语法规则（此规则确保可以恢复成为最初的轨迹）来去除其中的冗余。此轨迹可以包含循环和 back edges。SEQUITUR 是一个即时算法，这意味着当轨迹生成时，SEQUITUR 就可以同时对轨迹进行处理，这就不需要将整个轨迹记录下来，我们只需要对保存下来的 SEQUITUR 语法规则进行处理就可以了。

SEQUITUR 算法为：输入 执行轨迹 Tr，输出 语法 G，循环查找 S 中的每一个标记，将标记追加到规则 T 的末尾。在 G 过程中，注意处理 G 中有匹配的规则和重复的连字进行替换问题。

算法通过在规则 T 后追加标记、搜索结果规则和所有其它的冗余规则集来构建语法规则。如果追加之后的压缩轨迹的尾部与规则集中有匹配就进行相应的替换。如果判断出新的冗余，就将一个新的规则加入规则集并进行替换。每一次的替换产生时，算法也通过检查来保证成果规则被使用了超过一次。如果某规则只使用了一次，此规则会被替换回来，而且相应的规则也会从语法规则中移除。

在我们的例子中，我们应用这个算法来处理如图 3 所示的若干个轨迹连接起来的复合执行轨迹，其中，每一个调用和返回都用标记表示出来。每一个单独的轨迹由特殊符号 x 结束，当程序执行到 exit 时产生。对 SEQUITUR 算法的操作举例说明，我们考虑图 3 所示的轨迹。规则 T 最初是空的：

$T\rightarrow$

// 使用图 3 中间的程序轨迹，算法开始追加符号，以 A 开始，不断添加，直到发现重复的连字。

T$\rightarrow ACGrrBFIrHrr$

// 每一个步骤算法都检查，看看串最后的两个标记（连字）是不是重复出现的。在这个例子中，rr 出现了两次，因此，算法构建了一个语法规则并把它应用到 T 中。

$T\rightarrow$　$ACG1BFIrH1$

$1\rightarrow rr$

// 接下来继续添加，得到 $T\rightarrow ACG1BFIrH1rr$，它的尾部与 $1\rightarrow rr$ 相匹配，因此，rr 被替换，得到 $T\rightarrow ACG1BFIrH11$。

应用了新的规则之后，算法继续追加标记、查找冗余。在这个例子中，另外一个冗余 AC 被发现，因此在规则集中添加第二个规则。

$T\rightarrow 2G1BFIrH11x2$

$1\rightarrow rr$

$2\rightarrow AC$

当所有的标记被处理完成后语法规则就生成了，最终的结果如下：

$T\rightarrow 2G1BFIr53D1r3F5x$

$1\rightarrow rr$

$2\rightarrow AC$

$3\rightarrow x2$

$4\rightarrow H1$

$5\rightarrow 41$

在处理完成后，得到的结果语法中包含端点（函数名称、r 和 x）和规则（可包含端点和其它规则）。

算法 SEQUITUR 的时间复杂度是 $O(N)$，N 是轨迹的长度。被压缩轨迹的长度最坏情况（没有可行的压缩）下是 $O(N)$，最好情况下是 $O(\log N)$。包含循环的程序可能产生极端的长轨迹，但是循环将产生冗余，而这正是 SEQUITUR 算法所擅长的。

SEQUITUR 语法可以以全路径 DAG 图的形式被存储。全路径 DAG 图中的每一个节点就是 SEQUITUR 语

法中的一条规则。通过将每一个成果规则连接到他们引用的规则或非终端节点，可以构建一个直接非循环图。以上的SEQUITUR语法所产生的全路径*DAG*图如图4所示。

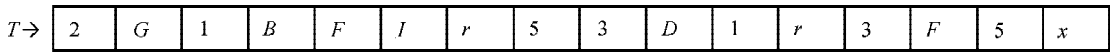

图4 全路径DAG图

Fig. 4 Whole path DAG

3.2 路径影响算法

用来评估影响的WPDAGImpact 算法为：{输入 语法 G ； 输入 被修改的过程 n；输出 影响过程集 S；structure Action}。进一步细化为：

{int skip = initially 0；int retn = initially 0；int stat = initially 0 ； Action act ； boolean fwd = initially true ； boolean bwd = initially true}

集合 S 为潜在影响过程集，初始为空，如果节点 n 存在于 G 中就将 n 加入到 S 中。利用search(n，p，*fwd*，*bwd*，*act*)循环查找包含 n 的每一个节点 p 。

我们可以从DAG图中的某一个节点开始，向前或向后以递归的形式顺序遍历每一个节点，直到遇到轨迹终结符号。以这种方式遍历WPDAG产生的影响集与遍历未压缩轨迹得到的影响集是一致的。

任何被修改的过程都有一个端点，WPDAGImpact就从这个端点开始，在WPDAG中向前搜索在被修改过程之后被调用的过程，向后搜索返回的过程。整数型act.stat 用来标记执行轨迹的结束，并相应地设置布尔值fwd和bwd。search 函数中用到了 fwdsearch 和bwdsearch两个函数来检索出语法中应该被加入到影响集 S 中的端点。

当向前检索及当管理向后检索返回的过程时，整数型act.retn和act.skip执行必要的簿记功能。fwdsearch匹配过程名和返回，然后将未匹配的数目通过结构act中的变量retn传递给bwdsearch。

bwdsearch 通过 act.retn 来确定在影响集中会包含多少个过程。当bwdsearch 增加了和act.retn数目相当的过程，向后的检索就可以停止了。Bwdsearch同样必须进行过程名和返回的匹配，因为在被修改的过程被执行之前就已经返回的过程是必须排除的。变量skip用来控制这一点：当遇到return时，skip就增加一；当遇到过程名时，skip就减一。当skip 的值为零时，过程就被添加到影响集 S 中。

由于SEQUITUR语法生成的WPDAG是没有循环的，因此WPDAGImpact算法中的遍历一定有终点。要确定WPDAGImpact算法对于一个未压缩的轨迹也是正确的并不难。

4 结语

本文通过对基于依赖的效费比折衷的影响技术和基于全路径的动态影响进行分析，提出了一种新的基于WPDAG的性价折中的回归测试选择方法，它能有效地提高测试选择精度，降低测试选择成本。

参考文献：

[1] 黄锡滋. 软件可靠性、安全性与质量保证[M]. 北京：电子工业出版社，2002：22-46.

[2] Rothermel G，Harrold M J. A safe, efficient regression test selection technique. ACM Trans[J]. On Software Engineering And Methodology，1997， 6(2)：173-210.

[3] Turver R J，Munro M. Early impact analysis technique for software maintenance[J]. Journal of Software Maintenance：Research and Practice，1994，6(1)：35-52.

[4] Bohner S，Arnold R. Software Change Impact Analysis[M]. Los Alamitos：IEEE Computer Society Press，1996.

[5] Nevill-Manning C, Witten I. Linear-time, incremental hierarchy inference for compression[C]// In Proc Data Compression Conference (DDC 97). [s.l.]：[s.n.]，1997：3-11.

[6] Larus J. Whole Program Paths[C]//In Proc. SIGPLAN PLDI 99. Atlanta：[s.n.]，1999：1-11.