一种分析和理解程序的方法——程序切片

李必信 郑国梁 王云峰 李宣东

(南京大学计算机软件新技术国家重点实验室 南京 210093)

(南京大学计算机科学与技术系 南京 210093)

摘 要 程序切片是一种分析和理解程序的技术,是通过对源程序中每个兴趣点分别计算切片来达到对程序的分析和理解.程序中某个兴趣点的程序切片不仅与在该点定义和使用的变量有关,而且与影响该变量的值的语句和谓词以及受该变量的值影响的语句和谓词有关.文中详细阐述了程序切片技术的研究与进展情况,并对目前存在各种程序切片方法和工具进行了比较;简单介绍了文中提出的面向对象的分层切片方法及其算法的思想;最后分析了程序切片技术目前还存在的一些问题及其发展趋势.

关键词 数据依赖,控制依赖,依赖图,程序切片,分层切片

中图法分类号 TP311.5

AN APPROACH TO ANALYZING AND UNDERSTANDING PROGRAM——PROGRAM SLICING

LI Bi-Xin, ZHENG Guo-Liang, WANG Yun-Feng, and LI Xuan-Dong

(*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

(*Department of Computer Science & Technology, Nanjing University, Nanjing* 210093)

Abstract Program slicing is a static analysis technique that extracts all statements relevant to the computation of a given variable. Program slicing is useful in program debugging, software maintenance, and program understanding. Program slices can be used to reduce the effort in examining software by allowing a software auditor to focus attention on one computation at a time. In this paper, the research and development of program slicing is discussed, all kinds of program slicing methods are compared, the progress made in object-oriented program slicing domain is also presented, and current problem and future directions are analyzed at the end of the paper.

Key words data dependence, control dependence, dependence graph, program slicing, hierarchy slicing

1 引 言

程序切片的概念由 Mark Weiser 在他的博士论文[1]中首次提出的.Weiser 声称一个切片与人们在调试一个程序时所做的智力抽象相对应.程序切片有两类定义方式:①Weiser 定义的切片是一个可执行的程序,

原稿收到日期:1999-06-03;修改稿收到日期:1999-09-20.本课题得到合肥经济技术学院科研基金(项目编号 99-183)和江苏省自然科学基金项目(项目编号 BK99038)资助.**李必信**,男,1969年生,博士研究生,主要研究方向为面向对象支撑技术及其环境和工具、程序理解等.**郑国梁**,1937年生,教授,博士生导师,主要研究方向为软件工程、面向对象技术.**王云峰**,男,1964年生,博士研究生,主要研究方向为面向对象技术、形式化技术.**李宣东**,男,1963年生,博士,副教授,主要研究方向为面向对象技术、形式化技术等.

是通过从源程序中移去零条或多条语句来构造的;②另一种定义是由程序中语句和控制谓词组成的一个子集,这些语句和控制谓词直接或间接影响在切片准则计算的变量的值,这类切片不必构成可执行的程序.计算程序切片时,首先一般要考虑以下几种情况:

(1) 是计算静态切片还是动态切片?对程序的某个变量来说,如果计算它对应的静态切片,则切片计算出来的该变量的值与原来的程序在任意的输入下执行时计算出的该变量的值相同;如果计算它的动态切片,则由切片和由原来程序计算出的该变量的值在某个输入下是相同的。实际上,静态切片考虑了程序的所有可能的执行路径,动态切片只考虑对于某个具体输入下程序实际执行的路径。静态程序切片是通过分析程序的源代码来获得程序的有关信息,是一种不包括人机交互作用的在编译时确定的方法。而动态程序切片记录多次程序运行的测试结果,并使用用户的实际输入产生精确的数据流信息(即在程序的某个特定执行过程中产生的依赖;

(2) 是计算前向切片还是后向切片?如果计算前向切片,则切片集合中包含了程序中所有能够影响变量在切片准则的值的部分;如果计算后向切片,则关心的是所有受切片准则的变量的值影响的程序部分;

(3) 是计算过程内切片、过程间切片还是模块间切片?过程内切片只考虑一个过程内部的语句,它不能衍生在过程调用存在的情况下的有效信息;过程间切片必须从过程调用的位置到调用的目标之间模型化参数传递;模块间切片能够越过模块边界.且允许使用库的切片和对不完全程序的切片;

(4) 是计算粗粒度切片还是细粒度切片?因为可以在不同粒度层次上计算程序切片.另外,计算程序切片时,还需考虑数据依赖和控制依赖,因为它们是切片算法的基础.

本文其他部分的安排如下:第2节简单介绍几种切片准则;第3节分别重点讨论了两种具有代表性的切片算法;第4节简要介绍和比较了几种流行的程序切片工具;第5节简要介绍我们提出的分层切片算法的思想;第6节讨论程序切片技术目前存在的问题、发展趋势并给出本文的结论.

2　程序切片准则

计算程序切片还必须考虑切片准则,对同样的程序计算切片时,如果切片准则发生变化,则计算得到的程序切片也会不同.因而程序切片准则是程序切片技术的重要环节.最常见的几种切片准则有:

静态后向切片准则:构造一个程序 P 的静态后向切片时,考虑的切片准则是一个二元组 (V,p),其中 V 是一组变量,p 是程序中的一个兴趣点.程序 P 的一个切片是任何一个在 V 中 p 点对变量具有与 P 相同影响的程序;

动态后向切片准则:构造一个程序 P 的动态后向切片构造时,考虑的切片标准是一个三元组 (V,p,x),其中 V 是变量的集合,p 是程序 P 中的一个兴趣点,x 是一个输入序列.当用 x 输入时,一个动态切片保留 P 的与 p 点的变量集合 V 有关的投影含义(project meaning);

条件切片准则:构造一个程序 P 的条件切片时,考虑的切片准则是一个四元组 (I,p,π,V),这里 I 是变量名的集合,这些变量的值是从输入序列中获得的,p 是程序 P 中一个兴趣点,π 是一个谓词,它的自由变量是 I 的一个子集,V 是一个变量名的集合.构造一个程序 P 的条件切片 S,当在一个满足 π 的初始状态执行时,条件切片 S 必须保持程序 P(与 V 有关)的投影含义.

另外,还有迭代切片准则、传统的前向切片准则、分解切片准则、多点切片准则、广义切片准则、无定型切片、对象切片、类层次切片、对象的状态切片和行为切片等切片准则,大家可以参考文献[2]~[5].

3　程序切片算法

程序切片的算法主要有 Weiser 的基于数据流方程的算法[1]、无定型切片算法[2]、Bergeretti 的基于信息流关系的算法[6]、K. J. Ottenstein 和 L. M. Ottenstein 以及 Horwitz 的基于程序依赖图的图形可达性算法[7,8]、杨洪的基于波动图的算法[9].另外,还有参数化程序切片算法[10]、并行切片算法以及面向对象的分层切片算法等.这里重点介绍基于数据流方程的算法和基于依赖图的算法.

3.1 基于数据流方程的算法

Weiser 是根据数据流方程的迭代解最早定义程序切片的.他定义的程序切片是通过对初始程序删除零条或多条语句而获得的一个可执行的程序.切片准则由一个二元组(n,V)构成,其中n是程序的CFG中一个结点,V是程序中变量的一个子集.关于切片准则(n,V)的程序切片是程序P的语句的一个子集S,它必须满足下列特性:

① S必须是一个有效程序;

② 对一个给定的输入,P发生中断时,S也会发生中断.无论何时与结点n相关的语句被执行时,计算V中所有变量的值(对P和V来说)分别是相同的.

公理 1. 对应任何一个切片准则,至少存在一个切片:程序本身.

如果对同一个切片准则来说,没有其他的切片比某个切片S_{minimal}包含更少的语句,则称S_{minimal}为最少语句的切片.Weiser 认为最少语句切片不一定是唯一的,且如何确定最少语句切片问题是不可判定的.

Weiser 描述了一个用来计算最少语句切片近似值的迭代算法.这种算法使用两种不同的迭代层次,即

① 跟踪可传递的数据依赖层。这要求迭代过程是在循环存在情况下发生的;

② 跟踪控制依赖层。该层把跟踪到的新的控制谓词控制的语句包含到某个控制谓词的切片中.对每个这样的谓词,重复步骤 1 直到把它依赖的所有语句包含进来为止.这类算法确定相关变量(relevant variable)的连贯(consecutive)集合,由此可得到相关语句的集合.

Weiser 迭代算法的第 1 层要确定直接相关变量:这是上面论述的迭代过程步骤 1 的一个实例.CFG 中结点i的直接相关变量的集合记为$R_C^0(i)$.迭代开始于初始值$R_C^0(n)=V$,且对任何结点$m\neq n$,满足$R_C^0(m=\varnothing$.图 1 中定义的一组方程可确定在 CFG 的边$i\rightarrow_{\text{CFG}} j$的终点$j$的相关变量的集合是如何影响在开始点$i$的相关变量的集合.这个过程的最小不动点就是在结点$i$的直接相关变量的集合.从$R_C^0$可得到直接相关语句集合$S_C^0$.图 1 表示了$S_C^0$是如何被定义为所有结点$i$的集合,$i$定义了变量$v$,该变量在$i$的一个 CFG 后继是相关的.

对 CFG 中的每条边 $i\rightarrow_{\text{CFG}} j$:

$$R_C^0(i)=R_C^0(i)\cup\{v\mid v\in R_C^0(j),\ v\notin Def(i)\}\cup\{v\mid v\in \mathrm{Ref}(i),\ Def(i)\cap R_C^0(j)\neq\varnothing\}$$

$$S_C^0=\{i\mid Def(i)\cap R_C^0(j)\neq\varnothing, i\rightarrow_{\text{CFG}} j\}$$

图 1 确定直接相关变量和语句的方程

Weiser 迭代算法的第 2 层要考虑控制依赖.如果在一个 if 或 while 语句控制谓词的控制体中至少有一条语句是相关的,则变量引用是间接相关的.为了实现这个目标,一条分支语句b的影响范围(range of influence)$Infl(b)$定义为由所有控制依赖b的语句组成的集合.图 2 表示分支语句的一个定义B_C^k.由于它们对S_C^k中结点i的影响,因而是间接相关的.间接相关变量的(indirectly relevant variable)$R_C^{k+1}(i)$得到确定.除$R_C^k(i)$中的变量之外,$R_C^{k+1}(i)$包含的变量是相关的,因为它们对B_C^k中的语句存在可传递的数据依赖.这通过再一次执行第 1 层迭代来确定(也就是跟踪可传递的数据流),同时考虑的一组准则是$(b,Ref(b))$,这里b是B_C^k中的一条分支语句(参见图 2).图 2 还定义了在$k+1$次迭代过程中间接相关语句(indirectly relevant statements)的集合S_C^{k+1}.这个集合由B_C^k中的结点i一起构成,结点i定义了一个变量,该变量对 CFG 的后继j来说是R_C^{k+1}相关的.由此可得下面的定义.

$$B_C^k=\{b\mid \exists\ i\in B_C^k, i\in Infl(b)\}$$

$$R_C^{k+1}(i)=R_C^k(i)\cup\bigcup_{b\in B_C^k} R_{(b,\mathrm{Ref}(b))}^k(i)$$

$$S_C^{k+1}=B_C^k\cup\{i\mid Def(i)\cap R_C^{k+1}(j)\neq\varnothing, i\rightarrow_{CFG} j\}$$

图 2 确定间接相关变量和语句的方程

定义 1. 程序切片是由程序中某个兴趣点的相关语句集合的不动点构成的集合.计算程序切片就是计算相关语句集合的不动点.

集合R_C^{k+1}和S_C^{k+1}分别是程序的变量和语句的非递减子集;计算得到的集合S_C^{k+1}的不动点就构成期望的程序切片.

Lyle 虽然提供了 Weiser 切片算法一种修改版,但除了在术语方面的细小改变外,与 Weiser 算法本质上没有什么两样.

Hausler 用函数的风格重述了 Weiser 的算法. 对每种类型的语句(空语句,赋值语句、语句组合,if 和 while),他定义了两个函数 δ 和 α. 粗略地说,这些函数分别表示一条语句是如何转换相关变量的集合 R_c,以及相关语句的集合 S_c. 函数 δ 和 α 是以一种组合的方式定义的. 对空语句和赋值语句,δ 和 α 可以按语法制导的方式从语句中产生出来. 语句序列和 if 语句的 δ 和 α 函数可以分别从它们构件的 δ 和 α 函数推导出来. 通过把它有效地转换成一个 if 语句的无穷序列来获得 while 语句的函数.

3.2　基于依赖图(DG)的图形可达性算法

Ottenstein 等在 1984 年提出了切片准则由一个程序点 p 和一个在 p 定义的或使用的变量 v 构成[7]. 且在一个程序依赖图(PDG)上使用图形可达性算法来计算切片. 根据他们对切片含义的理解,切片是由程序中所有可能影响 v 在 p 点值的语句和谓词构成. 1990 年,Horwitz 等在文献[8]中使用依赖图来计算切片,他们开发了一种过程间的程序表示——系统依赖图(SDG),并实现了一个基于系统依赖图的两阶段图形可达性切片算法. 由于使用调用位置的可传递的依赖流信息中包含被调用过程的上下文环境,所以该算法的计算精度比以前的算法更高. 后来,为了切片面向对象的程序又出现了多种依赖图,具有代表性的有:Zhao 提出的一种动态面向对象的依赖图(DODG)和系统依赖图网(SDN),分别用于解决面向对象程序的动态切片问题和并发面向对象的程序切片问题[11,12];Krishnaswamy 提出一种面向对象的程序依赖图(OPDG),并在 OPDG 上实现了计算语句切片的算法[13]. OPDG 是通过对传统的程序依赖图进行面向对象的扩充获得的,但 OPDG 不能表示虚继承、动态对象等问题,因而 OPDG 是不完全的面向对象程序表示. 下面简单分析一下各种依赖图在切片算法中应用及其特点.

(1)程序依赖图(PDG)

PDG 是程序的一种图形表示,它把控制依赖和数据依赖包含在单个的结构中. 如果给定程序中的语句 X 和 Y,则 X 和 Y 可以通过控制流或者数据流来彼此关联. 如果从 Y 至少可以引出两条路径,其中一条总会导致 X 的执行,而另一条可能导致 X 不执行,则称语句 X 控制依赖于语句 Y;如果有一条从 Y 到 X 的路径,且存在一个在 Y 点定义在 X 点使用的变量,且该变量在沿从 Y 到 X 的路径上其它任何地方没有被重新定义,则称语句 X 数据依赖于语句 Y. PDG 的形式定义是由一个控制依赖子图,一个控制流图和一个数据依赖子图组成. 其中 CDG 包含了程序中的控制依赖; CFG 描述了一个程序中的控制流,它类似于正常情况下的流图;DDG 是一个程序中语句之间所有数据依赖的集合. CDG 包含几种类型的结点,即语句结点表示程序中的语句;域结点概括了域中语句间的控制依赖;谓词结点(由此可以引出两条边)表示程序中的策略或分支条件. CDG 中的域结点可利用相同的控制依赖来集合语句. DDG 包含语句间的数据依赖. 可通过在 CDG 的结点之间插入数据依赖边来构造 DDG. 同样 PDG 不允许过程间分析,它也没有能力表示继承、多态性和动态定连等重要的面向对象的特性.

(2) 系统依赖图(SDG)

SDG 是程序的一种语法分析树表示. 结点表示程序构造(constructs),输入输出参数(in, out 参数)和调用位置等. 边表示与之相连的结点之间的各种依赖(例如数据依赖,控制依赖和声明等). 程序依赖图通常是为单个的过程间程序来定义的. 由于现实世界的程序一般是由多个过程组成的,所以必须考虑使用一种与现实世界程序匹配的表示方法,于是 Horwitz 等人提出了系统依赖图[8]. SDG 是一棵经过装饰的表示程序的语法分析树. 形式地说,SDG 是由一个程序依赖图和一组过程依赖图(PrDG)构成的有向、带标记的多重图. PDG 模型化软件系统中的主程序,PrDG 模型化软件系统中的多个过程体. SDG 可以用来处理过程间的数据流和控制流,并能表示参数传递. SDG 允许过程间分析,但面向对象程序远不是一些过程或方法的简单组合,所以 SDG 提供的机制还不足以描述这些面向对象的概念.

(3) 面向对象程序依赖图(OPDG)

OPDG 是由 Krishnaswamy 提出的一种表示面向对象程序的依赖图[13]. OPDG 由类层次图(CHG)、控制依赖子图(CDG)和数据依赖子图(DDG)3 个基本层次组成. CHG 包含每个类的类首部顶点和定义在类中的每个方法的方法首部顶点. CHG 中的边把每个类首部顶点和与其具有继承关系的类的相应的类首部顶点

连接起来. 由方法首部表示的方法顶点被连接到定义该方法的类的类首部顶点. 子类中没有重新表示从超类中继承的方法,因而消除了对继承方法的重复表示. 因为 CHG 不表示方法的任何实现细节,故不是一个程序的完全的静态表示. 这个阶段的 CHG 表示基本类设计以后的程序. CDG 中包含每个方法的实现. 因为这是一种静态表示方法,某些信息还不能够完全得到解决. 例如,就不能唯一地确定哪些消息的接收者将是动态绑定的. 这种表示会识别具有多态性的类的集合,而不是形式地定义接收者. 这是一种完全的静态表示,它能够支持许多分析. DDG 层中包含对象. 这使得把消息动态绑定到对象中的特定方法得到完全解决. 其它传统类型的数据依赖信息也得到表示,并为许多静态和动态分析提供支持. 这层表示为不同的应用(如调试、动态切片和其它分析)提供信息.

(4)动态面向对象依赖图(DODG)

DODG 是 Zhao 为解决面向对象的动态程序切片而采用的一种程序表示方法[11,12]. DODG 是一种有向图,可以清晰地表示某个面向对象程序的一个特定执行的语句实例之间的各种动态依赖. 并基于 DODG,提供了一种两阶段算法用于计算面向对象程序的动态切片.

(5)系统依赖网(SDN)

为了解决并发面向对象程序静态切片问题,Zhao 提出系统依赖网(SDN)的表示方法[11,12]. SDN 把以前的程序依赖表示扩展以后用来表示并发面向对象程序. 并发面向对象程序的一个 SDN 由一些表示每个主要过程、自由准则过程或程序中某个类的一个方法,以及用一些附加的弧来表示一个调用和被调用过程/方法之间的直接依赖,以及可传递的过程间数据依赖. SDN 不仅表示面向对象特性,还可以表示并发面向对象程序的并发问题. 一旦用 SDN 表示了并发面向对象程序,基于 SDN 的程序切片可以通过 SDN 上简单的顶点可达性问题来计算得到.

(6)增强的面向对象系统依赖图(EOOSDG)

EOOSDG 是我们对传统的系统依赖图(SDG)的一种面向对象扩充,它是一个比较完整的面向对象程序的系统依赖图表示方法. 通过引入多态性调用边、多态性选择边表示面向对象程序的多态性问题,通过引入虚继承边处理虚继承问题. EOOSDG 分别对方法、单个的类、相互作用的类、类层次以及完全的面向对象程序给出系统依赖图表示. 我们在 EOOSDG 上实现了两阶段图形可达性算法.

另外,还有基于信息流的算法思想;基于语句波动图的算法思想;John Field 等提出的参数化的通用程序切片算法;Mark Harman 等提出的无定型切片概念与准则;LI Bixin 等提出的面向对象程序的分层切片思想及算法,本文第 5 节中对分层切片有较详细介绍.

4 程序切片工具(PST)

目前,实用的程序切片工具和环境很多[14,15],按照切片语言的不同可分为以下几类:

(1)支持 C 语言的 PST

Wisconsin 程序切片工具 version 1.1 是一个支持对 C 程序操作的软件系统. 它包括后向切片、前向切片和消片 3 种功能,他们能够帮助用户获得对一个程序正在做什么和如何做的理解. 该切片工具由用来建立和操作控制流图和程序相关图的一个包构成,和分析 C 程序语法的一个前端类似,也把它们转换成适用于切片的中间表示;

(2)支持 Ada 语言的 PST

PSS/Ada 系统是一个 Ada 程序的静态切片的生成系统. 它以对 Ada 程序进行静态程序依赖性分析(包括数据依赖分析和控制依赖分析)为基础,分析了 Ada 程序中语句间存在的波动效应(ripple effect),并生成特殊的 Ada 程序切片,该切片是从 Ada 程序中抽取出的某些语句以相同的顺序重新组成的一个新程序段. PSS/ Ada 系统是一个 Ada 软件的测试、排错、分析、理解和维护的工具,在 Ada 程序的并行性检查、波动性分析、复杂性度量等方面也提供一定范围的支持. PSS/ Ada 系统作为对 Ada 程序设计支持环境 APSE 的工具集的扩充,丰富了开发 Ada 软件时的 Ada 环境;

(3)支持 Oberon-2 语言的 PST

The Oberon Slicing Tool (OST) 是由 Christoph Steindl 开发一种计算 Oberon-2 程序的程序切片的工具,对程序没有限制,也就是说,程序可以使用结构化类型(记录和数据)、任何类型的全局变量、在堆上的对象;函数可以随机的边际效益;过程可以嵌套;由于类型绑定过程(方法)和过程变量(函数指针)的存在,程序可以是递归的和动态联编的;程序可以由模块组成.OST 切片的速度是相当快的(几秒钟之内就可计算切片信息,一点也不耽误计算切片的时间).OST 具有下列主要特点:①OST 使用静态切片(能产生比动态切片更大的切片而效率却更高;②可以计算过程内、过程间和模块间切片;③既可以处理过程程序,也可处理面向对象程序;④OST 是以面向表达式的方式来计算切片(比面向语句的切片粒度更细);⑤使用用户反馈来限制化名和动态联编的影响;⑥使用一个仓库来储存已计算的切片信息,当输入已经切片的模块时要重用这些信息;⑦实现了模型-视图-控制器概念 (也就是说可以从多个视图来观察保持一致性相同切片;

(4)支持 C++/Java 语言的 PST

我们正在研制开发基于分层切片算法的面向对象程序切片工具 OOPST,它能够对静态类型的面向对象语言(如 C++,Java 等)进行分层切片.也就是从逻辑上把 C++/Java 软件分成程序、类结构、类(或对象)、过程(或方法)等不同的层次,在每层上分别实施过程间切片或过程内切片.这种办法降低了切片算法的复杂性,提高了切片程序的效率.

另外,GrammaTech 是一种为商业软件开发者、政府机构、著名大学提供的一种创新的软件开发工具.GrammaTech 产品是基于语言的,这意味着它需要理解特定程序设计语言的规则和结构.这导致 GrammaTech 工具可以使许多任务自动化(而大多数工程师还在用传统的基于文本的工具来手工完成这些任务);The Linz Oberon Slicing System 是 Johannes Kepler 大学的系统软件组开发的基于语言 Oberon 的模块间切片工具; Microsoft's slicing tool 是 Microsoft Research 程序分析组开发的一种程序切片工具等,有兴趣的读者可参见相关文献.

5　分层切片思想及其算法简介

我们在研究静态类型面向对象的程序切片时,提出了一种分层切片的思想,该思想由 2 个部分组成:

(1) 把面向对象的软件按逻辑结构分为 4 个层次,提出面向对象程序的层次模型,该模型由 4 个层次组成,即 OO 程序层表示整个程序,它包含全局变量、自由标准过程、单个类以及类层次结构几个方面:类层次结构层表示通过继承关系或通信关系形成的一组相互作用类;类或对象层表示单个的类或对象;过程/方法分别表示自由标准过程和类中定义的方法(又称成员函数)(如图 3);

(2) 在层次模型的基础上进一步实现计算面向对象程序切片的逐步求精算法.为了计算面向对象程序的切片,逐步求精算法采用的具体步骤就是:①OO 程序层:在计算与某个切片准则有关的程序切片时,把全局变量、自由标准过程、单个类以及类结构等都作为同等的单个实体来考虑.也就是说,如果一个类结构中某个类的某个方法中的某条语句影响切片准则或受切片准则影响,则把该类包含到切片集合中.否则,在进行下一步切片时不予考虑;②类层次结构层:进一步确定该类结构中影响切片算法准则的类或对象,并把不影响切片准则或不受切片准则影响的类或对象从类结构中删除;③类或对象层:进一步确定类或对象中影响切片准则或受切片准则影响的变量、语句和方法,删除那些与切片准则无关的变量、语句和方法等;④过程/方法层:进一步确定过程或方法中与切片准则有关的变量和语句,删除那些与切片准则无关的变量、语句和控制谓词等.

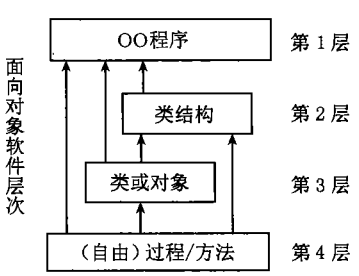

图 3　对象程序的层次模型

6　结　　论

程序切片是一种程序分析和理解技术,它的发展基本成熟,在理论和应用两方面的研究都取得了可喜的进展,特别是在程序的调试、测试、分解和集成、软件维护等领域具有广泛的应用[16~23].但是程序切片技术也

还存在一些困难和问题,主要体现在以下几个方面:①程序切片的算法比较单一,从对结构化软件到非结构化软件到面向对象软件的切片算法,基本上还是采用基于依赖图的图形可达性算法。对面向对象程序来说,由于各种依赖图特别复杂,利用图形可达性算法(或数据流方法,或信息流关系)计算切片就显得非常复杂,且效率低下;②在实际应用过程中,程序切片主要用于调试和测试程序,至于程序分析、程序理解方面的研究不是很多,也不很到位;③实际可用的程序切片工具较少,特别是切片面向对象程序(如 C++/Java 程序)的工具就更加稀少,显然不能满足软件开发方法发展的需要;④人们对程序切片技术重要性的认识不够,很多人热衷于进行一些时髦领域的研究,而对这些基础性的研究却提不起兴趣;⑤由于新兴的各种软件开发方法的推广使用,对支撑环境的要求越来越高,程序切片作为支撑环境的一种技术同样面临着各种挑战和机遇,传统的计算程序切片的方法实际上已经不能用来计算并行面向对象程序、分布式面向对象程序等的静态切片或动态切片,所以需要寻找新的方法;⑥在面向对象程序分析过程中,动态切片技术的研究比较晚,目前除了少数学者做了一些这方面的研究外,理论和应用的研究都处在初始阶段.

尽管程序切片技术存在着这些问题,它的作用是巨大的,人们对程序切片的理论和方法研究也从未间断,并且随着软件开发方法的发展而不断得到改进和扩充. 在最近的研究中,Krishnaswamy 等人开发了面向对象的程序依赖图(OPDG)计算了面向对象的语句切片;Marry 等人利用增强的系统依赖图(SDG)计算对象切片,并成功地计算了面向对象软件的切片;Tip 等人利用 Rossie-Friedman 类层次框架解决了类层次切片的计算问题[16];Zhao 等人分别利用 DODG 和 SDN 实现了面向对象的动态程序切片以及并发面向对象程序切片的计算方法[11].

程序切片技术今后的发展需考虑以下几个方面:①研究更有效、更准确、更快速的计算面向对象程序切片的算法,降低现有算法的复杂度,提高程序切片的精度。特别是尽量避免构造复杂的系统依赖图,可以采用分层切片的思想设计各种算法;②开发用来计算面向对象程序切片的有效工具,最好能开发出能够切片多种语言的切片工具;③在 DODG 的基础上进一步研究面向对象程序的动态程序切片方法,并开发相应的切片工具;④研究基于其他新型语言的程序切片方法和工具,例如基于函数型语言的切片方法和工具、基于逻辑型语言的切片方法和工具、基于分布式语言的切片方法和工具等等.

致谢 在本文的写作过程中,IBM T. J. Watson 研究中心的 Frank Tip 博士提供了大量有关程序切片的资料,樊晓聪博士在论文布局和语言组织方面提出很好建议,在此一并表示感谢.

参 考 文 献

1 Weiser M. Program slicing: Formal, psychological and practical investigations of an automatic program abstraction method [Ph D dissertation]. University of Michigan, Ann Arbor, Michigan, 1979

2 Harman M, Sebastian Danicic. Amorphous program slicing. In: Fifth International Workshop on Program Comprehension, Dearborn, Michigan, 1997

3 Chen J L, Wang F J, Chen Y L. Slicing object-oriented programs. In: Proc of the APSEC'97, Hongkong, 1997. 395～404

4 Law R C H. Object-oriented program slicing[Ph D dissertation], University of Regina, Regina, Canada, 1994

5 Tip F, Choi J D, Field J, Ramalingam G. Slicing class hierarchy in C++. In: Proc of the 11th Annual Conf on Object-Oriented Programming, System, Language, and Application. 1996. 179～197

6 Bergeretti J F, Carre B A. Information-flow and data analysis of while-program. ACM Trans on Programming Languages and System, 1985, (7): 37～61

7 Ottenstein K J, Ottenstein L M. The program dependence graph in a software development environment. In: Proc of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, 1984. 177～184

8 Horwitz S, Reps T, Binkley D. Interprocedural slicing using dependence graphs. ACM Trans on Programming Languages and System, 1990, 12(1): 26～60

9 杨 洪, 徐宝文. PSS/Ada 程序切片系统的设计和实现. 计算机研究与发展. 1997, 34(3): 217～222
(Yang Hong, Xu Baowen. Design and implementation of a PSS/Ada program slicing system. Journal of Computer Research and Development(in Chinese), 1997, 34(3): 217～222)

10 Field J, Ramalingam G, Tip F. Parametric program slicing. In: Conf Record of the 22nd ACM Symposium on Principles of Programming Languages. San Francisco, CA: ACM Press, 1995. 379～392

11 Zhao J J. Dynamic slicing of object-oriented program. Information Processing Society of Japan, Tech Rep: SE-98-119. 1998. 17～23

12 Zhao J J, Cheng J D, Ushijima K. Static slicing of concurrent object-oriented programs. In: Proc of the 20th IEEE Annual International

Computer Software and Applications Conf. IEEE Computer Society Press, 1996. 312～320
13 Krishnaswamy A. Program slicing: An application of object-oriented program dependency graphs. Department of Computer Science, Clemson University, Tech Rep: TR94-108, 1994
14 Tommy Hoffner. Evaluation and comparison of program slicing tools. Department of Computer and Information Science, Linköping University, Tech Rep: LiTH-IDA-R-95-01, 1995
15 Reps T W. The Wisconsin program-integration system reference manual: Release 2.0. Computer Sciences Department, University of Wisconsin-Madison, 1993
16 Tip T. A survey of program slicing techniques. Journal of Programming Languages, 1995, 3(3): 121～189
17 Larsen L D, Harrold M J. Slicing object-oriented software. In: Proc of the 18th International Conference on Software Engineering, German, 1996
18 Andrea De Lucia. Anna Rita Fasolino, Malcolm Munro. Understanding function behaviors through program slicing. In: Proc of the Fourth Workshop on Program Comprehension, Berlin, Germany, 1996
19 Bogdan Korel, Jurgen Rilling. Application of dynamic slicing in program debugging. In: Proc of the Third International Workshop on Automatic Debugging (AADEBUG'97), Linköping, Sweden, 1997
20 Filippo Lanubile, Giuseppe Visaggio. Extracting reusable functions by program slicing. IEEE Trans on Software Engineering, 1997, 23(4): 246～259
21 Gallagher K B, Lyle J R. Using program slicing in software maintenance. IEEE Trans on Software Engineering, 1991, 17(8): 751～761
22 Bates S. Horwitz S. Incremental program testing using program dependence graphs. In: Proc of the Twentieth ACM Symposium on Principles of Programming Languages, 1993. 384～396
23 Jakson D, Rolins E J. A new model of program dependence for reverse engineering. In: Proc of the Second ACM SIGSOFT Conf on Foundations of Software Engineering, 1994. 2～10

第七届中国机器学习学术会议(CMLW'2000)征文通知

由中国计算机学会人工智能与模式识别专业委员会和中国人工智能学会机器学习专业委员会等单位联合主办,南京大学承办的第七届中国机器学习学术会议,将于 2000 年 10 月在南京大学举行. 欢迎国内各界专家学者研究人员踊跃投稿. 现将有关征文事宜通知如下:

一、征文范围

①机器学习、知识获取;②数据挖掘与知识发现;③人工神经网络、遗传算法;④模糊理论与技术、模糊神经网络;⑤计算智能;⑥多 Agent 系统学习;⑦人脑的智能活动及思维模型;⑧基于 CASE 的学习;⑨语音与图像处理和理解;⑩自然语言理解;⑪分类与识别;⑫学习的认识机制与模拟;⑬智能学习系统、机器学习应用;⑭其它有关机器学习的文章.

二、征文要求

1. 论文必须未公开发表过,一般不超过 6000 字;
2. 论文包括题目、作者姓名、作者单位、中、英文论文摘要、关键字、正文和参考文献;另附作者姓名、单位、地址、邮编、传真及 E-mail 地址;
3. 论文一律为 A4 打印稿,一式两份.(请采用 Word 排版);
4. 会议将出版论文集,打印清样格式在录取时另行通知;
5. 征文请寄:南京大学计算机科学与技术系　陈世福(邮编:210093)

三、关键日期

截 稿 日 期:2000 年 3 月 30 日
录 用 通 知:2000 年 4 月 30 日
清样付印日期:2000 月 30 日

承 办 单 位:南京大学计算机科学与技术系　　邮　　编:210093
联 系 人:陈世福　谢　琪　　联系电话:(025)3593163
E-mail: ChenSF@nju.edu.cn　　传　　真:(025)3300710

中国机器学习学会秘书处
南京大学计算机软件新技术国家重点实验室
南京大学计算机科学与技术系
1999.11.8