面向方面程序的动态语义研究

谢 刚[1] 蒋 强[1] 石 磊[2,3]
(贵州师范大学大数据与计算机科学学院 贵阳 550001)[1]
(北京邮电大学智能通信软件与多媒体北京市重点实验室 北京 100876)[2]
(北京邮电大学计算机学院 北京 100876)[3]

摘 要 目前,针对面向方面程序,许多研究者已定义了各种各样的形式语义。但是,没有一种语义能被软件设计者和开发者所理解。针对该问题,在已有研究的基础上,应用统一程序理论中的设计来定义面向方面的动态语义。同时,以一个例子来说明该语义的使用。

关键词 面向方面程序,动态,语义
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.08.032

Dynamic Semantics of Aspect-oriented Programming

XIE Gang[1] JIANG Qiang[1] SHI Lei[2,3]
(School of Big Data and Computer Science, Guizhou Normal University, Guiyang 550001, China)[1]
(Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China)[2]
(School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China)[3]

Abstract At present, many researchers have developed various formal semantics for aspect-oriented program. However, none of the semantics can be understood by software designers and developers. Based on the existing research, this paper defined a dynamic semantics of aspect-oriented programs through using the definition of design in unifying theories of programming. The approach was enumerated with a case to demonstrate the usage of the semantics.

Keywords Aspect-oriented programming, Dynamic, Semantics

1 引言

Kiczlaes 等[1]提出了一种面向方面程序的设计思想(以下简称 AOP)。可以看出,AOP 是 OOP 的扩充,并且是一种对横切关注点[2]进行模块化的方法。程序验证、程序编译等领域的研究人员对其产生了浓厚的兴趣。

目前,大量研究人员已开发出了各种 AOP 语言[3-9],AspectJ 是其中使用最为广泛的语言,对其语义进行研究极具价值。

何积丰等[10]指出,只有将软件开发与形式化方法统一起来,才能提高软件的可靠性。因此,亟需一种源语言层次的语义来描述软件行为,以便进行形式化验证。

目前,许多研究者已提出了各种各样的形式语义[11-22]。其中 Wand 等[11]为 AspectJ 中的通知和动态接入点定义了一种无类型的指称语义。Jagadeesan 等[12]为 AspectJ 的核心元素定义了操作语义,其中包括切点和通知的操作语义。Lämmel 等[13]为方法调用 interception 定义了大步语义,以对面向对象程序进行扩展。这些特征对于 AspectJ 建模是理想的。Walker 等[14]提出了一种比较简单的形式化模型,该模型将 λ 演算用于带标记的连接点和通知中。综上所述,以上工作的语义模型非常复杂,使得对系统进行证明十分困难。Tucker 等[15]认为范围演算标志(scope tag)能被使用在无类型的高阶功能语言中,以对静态语义和动态语义进行定义。Masuhara 等[16]使用一种通用模型来描述横切结构,并用 Scheme 对该模型的语义进行定义。Tabareau[17]用 λ_2 为 AOP 语言 MinAML 定义相应的语义。Molderez 等[18]为 AOP 语言 ContractAJ 定义操作语义。Zhang 等[19]使用产品家族代数(product family algebra)定义编织过程的语义。这些工作都是通过转换系统对面向方面程序的语义进行研究。文献[20]为 AspectJ 中的切点(pointcut)定义语义,文献[21-22]直接使用类 AspectJ 语言描述类声明和方面声明的语义,该语义虽然能被软件设计者和开发者所理解,但只能验证声

到稿日期:2017-12-01 返修日期:2018-03-19 本文受国家自然科学基金(61563011),贵州师范大学资助博士科研项目资助。
谢 刚(1980-),男,博士,副教授,主要研究方向为计算机软件与理论,E-mail:xiegang@gznu.edu.cn(通信作者);蒋 强(1993-),男,硕士生,主要研究方向为人工智能;石 磊(1986-),男,博士生,主要研究方向为社交网络挖掘与搜索、智能信息处理。

明部分的正确性。本文将在文献[20-22]的基础上定义面向方面程序的动态语义。

本文第 2 节对相关知识进行介绍;第 3 节对 AOP 的相关概念进行介绍;第 4 节为 AspectJ 定义一种动态语义;第 5 节通过一个示例来说明本文所提语义的正确性;最后总结全文,并对未来的研究进行初步探讨。

2 预备知识

2.1 逻辑形式推演规则[23]

本文用到的谓词逻辑形式推演规则如下。

规则 1 $C\wedge\exists xD(x)\Leftrightarrow\exists x(C\wedge D(x))$。

规则 2 $\exists x(C(x)\wedge D(x))\Rightarrow\exists xC(x)\wedge\exists xD(x)$。

规则 3 $C(t)\Leftrightarrow\exists x[x=t\wedge C(x)]$,其中 $C(x)$ 是通过在 $C(t)$ 中把 t 的某些出现替换成 x 而得。

2.2 谓词的顺序复合[24]

假设 A 和 B 是两个谓词,并且 $\mathrm{in}\alpha'B=\mathrm{out}\alpha A=\{v'\}$,即 A 的输出变量是 B 的输入变量,则:

$$A(v');B(v)=_{\mathrm{df}}\exists v_0\cdot(A(v_0)\wedge B(v_0)) \tag{1}$$

推论 1[24] $A;\mathrm{false}=\mathrm{false}$

2.3 重要符号的定义[10]

定义 1 对于任意的 $g:\{b\vdash\!\!\rightarrow x\mid b\in A,x\in B\}$,$d\in A$,$r\in B$,有:

$$g\oplus\{d\vdash\!\!\rightarrow r\}=_{\mathrm{df}}\begin{cases}g\cup\{d\vdash\!\!\rightarrow r\}, & \text{if } g(d)=\Phi\\(g-\{d\vdash\!\!\rightarrow y\})\cup\{d\vdash\!\!\rightarrow r\}, & \text{if } g(d)=y\end{cases}$$

例 1 假设 $D=\{L,D,X,A,B,C,E\}$,$f=\{L\vdash\!\!\rightarrow D,D\vdash\!\!\rightarrow X,A\vdash\!\!\rightarrow B\}$,则有:

$(f\oplus\{H\vdash\!\!\rightarrow G\})\oplus\{L\vdash\!\!\rightarrow Y\}=\{H\vdash\!\!\rightarrow G,L\vdash\!\!\rightarrow Y,D\vdash\!\!\rightarrow X,A\vdash\!\!\rightarrow B\}$

3 AOP 的相关概念

3.1 基本概念

定义 2(核心关注点) 系统要实现的主要功能和目标,如职工工资查询等[2]。

定义 3(横切关注点) 各个核心关注点间共享的功能,如安全。对于已实现的系统,横切关注点还包含对系统的更新需求[2]。

定义 4(连接点) 程序的执行过程中合适定义(well-defined)的点,如对构造方法的调用和对方法的执行[1]。

定义 5(切点) 一系列连接点的集合,定横切关注点需要插入的位置[1]。

定义 6(通知) 在切点所选定的连接点处要执行的代码[1]。

定义 7(方面) 对横切关注点进行封装的基本模块[1]。

3.2 面向方面程序的示例[22]

图 1 为 AspectJ 的程序代码,该程序由类 *Main*,*Point* 和方面 A 与 B 组成。其中,类 *Point* 是基本代码,包含成员变量 x,成员方法 *setX*,*getX* 和 *add*;类 *Main* 为主程序,是程序的入口,只包含一个类静态方法 *main*();方面 A 是横切代码,包含切点 p 和一个 *after* 通知,通知的作用是显示 *Point* 类的成员变量 x 的值;方面 B 也是横切代码,包含切点 p 和一个 *after* 通知,该通知的作用与方面 A 中通知的作用一样。

```
class Point{
  int x;
  public int getX(){
    return x;
  }
  public void setX(int x){
    this.x=x;
  }
  public void add(int y){
    this.x=this.x+y;
  }
}
class Main{
  public static void main(String[] arg s)
  {
    Point c=new Point();
    c.setX(100);
    c.add(5);
  }
}
aspect A{
  pointcut p(Point d):call(void Point.add (int))&&target(d)
  after(Point e) returning:p(e){
  System.out.println("x is changed to"+e.getX());
  }
}
aspect B{
  pointcut p(Point d):call(void Point.add (int))&&target(d)
  after(Point e) returning p(e){
  System.out.println("x is"+e.getX());
  }
}
```

图 1 AspectJ 示例代码

Fig. 1 AspectJ sample code

该程序的执行过程如下:首先,由主方法 main 创建 *Point* 的实例 c,实例 c 调用成员方法 *setX* 将成员变量 x 赋值为 100,接着调用成员方法 *add*,由于对 *add* 的调用与切点 p 所捕获的连接点匹配,而与切点 p 绑定的都是 *after* 通知,先调用 *add*,即 x 的值变为 105;然后,根据通知的执行规则,先执行方面 B 的通知,即在屏幕上显示"x is 105";最后,再执行方面 A 的通知,在屏幕上显示"x is changed to 105"。

4 语义

4.1 一种 AOP 语言

为了便于软件设计人员和开发人员理解语义,本文采用类似 AspectJ 的抽象 AOP 语言[21-22],该语言的 BNF 范式如表 1 所列。

表 1　AOP 语言的 BNF

Table 1　BNF of AOP

程序	prog::=cdecls;adecls;Main
方面声明序列	adecls::=(|adecl|adecl;adecls
单个方面声明	adecl::=aspect A[extends D]{adefs;mdefs;pcdefs;addefs}
切点声明序列	pcdefs::=pcdef|pcdef;pcdefs
单个切点声明	pcdef::=pointcut pc(prameters):pcd
原始切点描述符	pcd::=call(MethodPattern)|execution(MethodPattern) |get(FieldPattern)|set(FiledPattern) |call(ConstructorPattern)|execution(ConstructorPattern) |initialization(ConstructorPattern) |preinitialization(ConsstructorPattern) |withincode(MethodPattern) |withincode(ConstructorPattern) |this(T)|this(x)|target(T)|target(x) |pcd&&pcd|pcd‖pcd|!pcd
匹配模式	TP::=*|C+|C TyPattern::=TP|T|TyPattern‖TyPattern| TyPattern&&TyPattern|!TyPattern MethodPattern::= TyPattern TP.m(TS) ConstructorPattern::= TyPattern TP.new(TS) FieldPattern::= TyPattern TP.x|TyPattern x
通知声明序列	addefs::= addef|addef;addefs
单个通知声明	addef::=advicetype ad(prameters):pc(y){c} advicetype::=before|after|around
类声明序列	cdecls::= cdecl|cdecl;cdecls
单个类声明	cdecl::=visibity class C[extendsD]{adefs;mdefs}
属性声明序列	adefs::= adef|adef;adefs
单个属性声明	adef::=visib:T a[=l]
类的可见性	visibility::=private|public
属性的可见性	visib::=private|public|protected
类型说明符序列	TS::= T| T; TS
单个类型说明符	T::=int|float|string|char|Boolean|C
方法声明序列	mdefs::=mdef |mdef;mdefs
单个方法声明	mdef::=m(prameters){c}
主方法	Main::=(exts;c)
外部变量声明序列	exts::=ext|ext;exts
单个外部变量声明	ext::= T x[=l]
形式参数序列	prameters::=ext|ext; prameters
命令	c::=skip|chaos|le:=C.new(eS)|var T x[=e]|end x |le.m(eS) |c;c| c◁b▷c|c⊓c|b*c|les:=eS
表达式序列	eS=e|e;eS
单个表达式	e::=x|le|C(e)|l|f(eS) leS=le|le;leS le::=o|le.a|self l::=cn|null

4.2　一些重要概念的定义

定义 8[22]　一个连接点是一个五元组$\langle jpt,o,id,vs,returntype\rangle$,其中:

1)jpt 表示连接点的类型,即 $jpt\in JPT$,$JPT=_{df}\{pcall, pexecution, set, fget, finit, fpreinit, aexecution, pwithin, pwithincode\}$。

2)o 表示与方法或属性绑定的对象名,$o\in INDENTIFIER$,$INDENTIFIER$ 表示标识符构成的集合,下同。

3)id 表示方法名或属性名,$id\in INDENTIFIER$。

4)vs 表示方法的实际参数,即 $vs\in VAL^{*}\cup\varepsilon$,$\varepsilon$ 表示空序列,VAL 表示变量的集合[10],下同。

5)$returntype$ 表示方法的返回值类型,即 $returntype\in T\cup CNAME\cup\varepsilon$,$CNAME$ 表示声明的类的集合[10],T 表示变量类型的集合[10],下同。

定义 9[22]　一个切点 p 是一个五元组$\langle A,pc,PVT,PV,pcd\rangle$,其中:

1)A 表示切点所在的方面名,$A\in ASNAME$。

2)pc 表示程序中声明的切点名,$pc\in INDENTIFIER$。

3)PVT 表示切点的形式参数类型,即 $PVT\in(T\cup CNAME)^{*}\cup\varepsilon$。

4)PV 表示切点的形式参数名字,$PV\in INDENTIFIER$。

5)pcd 表示切点描述符,即 $pcd\in PCD$,PCD 表示所有基本切点描述符构成的集合。

例 2　$\langle pc,\varepsilon,\varepsilon,call(C.m())\rangle$表示切点($pointcut\ pc():call(C.m())$;$\langle pc,C,b,target(b)\rangle$表示切点($pointcut\ pc(C\ b):target(b)$)。

定义 10[22]　一个通知 adv 是一个七元组$\langle A,ad,advicetype,ADVT,ADV,pc,c\rangle$,其中:

1)A 表示切点所在的方面名,$A\in ASNAME$。

2)ad 表示通知的名字,即 $ad\in INDENTIFIER$。

3)$advicetype$ 表示通知的类型,即 $advicetype\in\{before, after, around\}$。

4)$ADVT$ 表示通知的形式参数类型,即 $ADVT\in(T\cup CNAME)^{*}\cup\varepsilon$。

5)ADV 表示通知的形式参数名字,即 $ADV\in VAR^{*}\cup\varepsilon$,$VAR$ 表示变量的集合[10],下同。

6)pc 表示程序中声明的切点名,即 $pc\in INDENTIFIER$。

7)c 表示通知体,与方法体类似,由一系列的命令构成[10]。

定义 11　假设 CJP 是从命令的子集到 $JoinPoint$ 的幂集之间的函数,其形式化定义如下。

$CJP:Cset\rightarrow 2^{JoinPoint}$,其中:

$Cset=_{df}\{le:=C.new(es),le.m(es_1;es_2),le.a:=e,x=f(le.a)\}$

$CJP(le:=C.new(es))=_{df}\{\langle pcall,le,new,es,\varepsilon\rangle,\langle pexecution,le,new,es,\varepsilon\rangle,\langle finit,le,new,es,\varepsilon\rangle,\langle fpreinit,le,new,es,\varepsilon\rangle\}$

$CJP(le.m(es_1;es_2))=_{df}\{\langle pcall,le,m,es_1,detype(es_2)\rangle,\langle pexecution,le,m,es_1,detype(es_2)\rangle\}$

$CJP(le.a:=e)=_{df}\{\langle fset,le,a,\varepsilon,\varepsilon\rangle\}$

$CJP(x=f(le.a))=_{df}\{\langle fget,le,a,\varepsilon,\varepsilon\rangle\}$

例 3　$CJP(a.add(2))=\{\langle pcall,a,add,2,int\rangle,\langle pexecution,a,add,2,int\rangle\}$,即命令 $a.add(2)$不仅可作为一个方法调用连接点,还可作为一个方法执行连接点。

4.3　设计

本文将动态语义定义为统一程序理论[24](Unifying Theories of Programming,UTP)中的设计。设计表示用户和程序之间的合同。该合同表示如果程序能在满足前置条件 p 的状态下开始,那么在满足后置条件 R 的状态下一定终止。

为了方便,采用 rCOS 定义的设计。该定义把设计定义成如下形式[10]:

$$\beta:p\vdash R=_{df}p\vdash(R\wedge\underline{w}'=\underline{w})\qquad(2)$$

其中，$\underline{w}$表示设计过程中不会发生变化的变量的集合，$\underline{w}\subseteq$ in α；β表示设计过程中会发生变化的变量的集合，$\beta\subseteq$in α；$w\cup\beta=$in α；$\underline{w}'\subseteq$out α。

在本文中，in $\alpha=\Omega$，out $\alpha=\Omega'=\{ASNAME', Advice', Pointcut', ANAME', superclass', superaspect', pricname', pri', prot', pub', op', pubname\}$[22]。

定理 1[10] $((p_1\vdash R_1);(p_2\vdash R_2))=((p_1\wedge\neg(R_1;\neg p_2))\vdash(R_1;R_2))$。

4.4 动态语义

面向程序的动态行为，实际上是根据一定的规则将方面代码织入到基本代码中，这个过程被称为方面编织。方面编织是面向程序的核心支撑技术，它分为动态编织和静态编织。其中，静态编织是指在编译时将 *advice* 和一些条件检查代码插入到程序中合适的位置；动态编织是指在程序运行过程中对连接点进行匹配并调用相应的通知。本文采用的是动态编织规则。

4.4.1 PCD 的语义[20]

面向方面程序在编织前首先要完成切点的匹配，因此本文使用文献[20]定义的匹配函数，该函数的定义如下。

定义 12 假设 *match* 是从 $PCD\times JoinPoint$ 到$\{0,1,x[o]\}$的函数，其中，*PCD* 表示切点描述符的集合[20]，*JoinPoint*[22]表示连接点集合，$x[o]$表示将对象 o 与变量 x 进行绑定。*match* 函数的形式化定义如下：

$$match(call(T\,C.m(TS)),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } jpt=pcall\wedge dtype(o)\leqslant C\wedge\exists N\cdot(m\in op(N)\wedge C\leqslant N)\wedge id=m\wedge dtype(vs)\leqslant TS\wedge returntype=T\\ 0, & \text{otherwise}\end{cases}$$

其中，$dtype(o)$表示对象 o 的声明类型，$op(C)$表示类 C 的方法名的集合。

$$match(execution(T\,C.m(TS)),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } jpt=pexecution\wedge type(o)\leqslant C\wedge m\in op(C)\wedge id=m\wedge dtype(vs)\leqslant TS\wedge returntype=T\\ 0, & \text{otherwise}\end{cases}$$

其中，$type(o)$表示对象 o 的类型。

$$match(get(T\,C.a),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } a\in ATTR(C)\wedge jpt=fget\wedge type(o)\leqslant C\wedge dtype(id)\leqslant T\wedge id=a\wedge type(vs)\leqslant dtype(a)\\ 0, & \text{otherwise}\end{cases}$$

其中，$ATTR(C)$表示 C 的属性。

$$match(set(T\,C.a),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } a\in ATTR(C)\wedge jpt=fget\wedge type(o)\leqslant C\wedge dtype(id)\leqslant T\wedge id=a\wedge type(vs)\leqslant dtype(a)\\ 0, & \text{otherwise}\end{cases}$$

$$match(initialization(C.new(TS)),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } jpt=finit\wedge type(o)\leqslant C\wedge id=new\wedge dtype(vs)\leqslant TS\wedge id=new\wedge returntype=\text{void}\\ 0, & \text{otherwise}\end{cases}$$

$$match(preinitialization(C.new(TS)),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } jpt=fpreinit\wedge type(o)\leqslant C\wedge id=new\wedge dtype(vs)\leqslant TS\wedge id=new\wedge returntype=void\\ 0, & \text{otherwise}\end{cases}$$

$$match(target(C),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } (jpt=pexecution\vee jpt=fget\vee jpt=fset\vee jpt=finit)\wedge type(o)\leqslant C\\ 0, & \text{otherwise}\end{cases}$$

$$match(target(x),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}x[o], & \text{if } (jpt=pexecution\vee jpt=fget\vee jpt=fset\vee jpt=finit)\wedge type(o)\leqslant dtype(x)\\ 0, & \text{otherwise}\end{cases}$$

$$match(this(C),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}1, & \text{if } (jpt=pexecution\vee jpt=finit)\wedge type(o)\leqslant C\\ 0, & \text{otherwise}\end{cases}$$

$$match(this(x),\langle jpt,o,id,vs,returntype\rangle)=_{df}\begin{cases}x[o], & \text{if } (jpt=pexecution\vee jpt=finit)\wedge type(o)\leqslant dtype(id)\\ 0, & \text{otherwise}\end{cases}$$

$$match(withincode(T\,C.m(TS)),jp)=_{df}\begin{cases}1, & \text{if } jp\in body(m)\wedge(jp.jpt=pcall\vee jp.jpt=fset\vee jp.jpt=fget)\\ 0, & \text{otherwise}\end{cases}$$

$$match(withincode(C.new(TS),jp))=_{df}\begin{cases}1, & \text{if } jp\in body(new)\wedge((jp.jpt=pcall\wedge jp.id\neq new)\vee jp.jpt=fset\vee jp.jpt=fget\vee match(execution(C.new(TS)),jp)\vee match(initialization(C.new(TS),jp))\vee match(preinitialization(C.new(TS),jp)))\\ 0, & \text{otherwise}\end{cases}$$

$match(pcd_1\,\&\&\,pcd_2,jp)=_{df}match(pcd_1,jp)\wedge match(pcd_2,jp)$

$match(pcd_1\parallel pcd_2,jp)=_{df}match(pcd_1,jp)\vee match(pcd_2,jp)$

$match(!pcd,jp)=_{df}5match(pcd_1,jp)$

$match(pcd,\{jp_1,jp_2,\cdots,jp_n\})=_{df}\vee_{i=1,2,\cdots,n}match(pcd,jp_i)$

且 *match* 函数必须满足以下规则：

1) $1\wedge x[o]=x[o]$；

2) $0\wedge x[o]=0, 0\vee x[o]=x[o]$；

3) $x[o_1]\wedge x[o_2]=x[o_1],x[o_2]$。

4.4.2 通知声明部分的语义

定义 13(通知声明的语义) 假设一个通知声明为 *addecl*，其语义如下：

$$[[addecl]]=_{df}\{Advice\}:\mathbb{D}(addecl)\vdash Advice'=Advice\cup\{\langle A,adv,advicetype,ADVT,ADV,pc,c\rangle\}$$

其中，$\mathbb{D}(addecl)$表示通知 *addecl* 是否合适定义[21]，*Advice* 为通知的集合[21]。

该定义的直观意义是：每定义一个通知，通知集合 *Advice* 就会增加一个元素。

当程序含有多个方面时,通知之间的关系可分为两种情况:一种是通知之间没有任何关系,另一种是多个同类型的通知与同一个切点绑定。当多个同类型通知与同一个切点绑定时,通知的优先级决定编织的顺序。由于不同的编译器对通知的优先级的确定是不同的,从而导致同一程序在不同的编译器中的运行结果不一样,使得程序难以移植和预测,该问题被称为通知的冲突。为解决该问题,本文引入同类型通知的复合运算,将这些通知合并成一个通知,其形式化定义如下。

定义 14　假设 adv_1 和 adv_2 是两个通知,如果 $adv_1.pc.pcd=adv_2.pc.pcd$ 且 adv_1 比 adv_2 先被定义,则它们的复合仍是通知,记作 $adv_1 \boxplus adv_2$,其定义如下:

$$adv_1 \boxplus adv_2 =_{df} \begin{cases} \{Advice\}:\mathbb{D}(adv_1)\wedge\mathbb{D}(adv_2)\vdash Advice'=Advice-\{adv_1,adv_2\}\cup\{\langle adv_1.A,adv_1.ad,adv_1.advicetype \\ ADVT_1,adv_1.ADV,adv_1.pc,adv_1.c;adv_2.c_2[ADV_1/ADV_2]\rangle\}, \\ \qquad\qquad \text{if } adv_1.advicetype=adv_2.advicetype\ after \\ \{Advice\}:\mathbb{D}(adv_1)\wedge\mathbb{D}(adv_2)\vdash Advice'=Advice-\{adv_1,adv_2\}\cup\langle adv_1.A,adv_1.ad,adv_1.advicetype,adv_1. \\ ADVT,adv_1.ADV,adv_1.pc_1,adv_2.c_2[ADV_1/ADV_2];adv_1.c_1\rangle\}, \\ \qquad\qquad \text{if } adv_1.advicetype=adv_2.advicetype=after \\ \{Advice\}:\mathbb{D}(adv_1)\wedge\mathbb{D}(adv_2)\vdash Advice'=Advice, \qquad \text{otherwise} \end{cases}$$

该定义的直观意义是:两个通知复合后,将后定义的通知体加入到先定义的通知体中,同时,将后定义的通知从方面中删除,即从 $Advice$ 中删除通知 adv_2。

该运算需满足如下性质:

性质 1(恒等律)　$adv \boxplus o=o \boxplus adv=adv$,其中 o 称为空通知,不会对基本程序产生影响。

性质 2(结合律)　$(adv_1 \boxplus adv_2)\boxplus adv_3 = adv_1 \boxplus (adv_2 \boxplus adv_3)$。

定义 15　假设 $addecls=(addecl_1;addecl_2;\cdots;addecl_n)$,其中 $addecl_i(i=1,\cdots,n)$ 表示第 i 个通知声明,其对应的通知的形式化定义为 adv_i。

$$[[addecls]]=[[addecl_1;addecl_2;\cdots;addecl_n]]$$
$$=adv_1 \boxplus adv_2 \boxplus\cdots\boxplus adv_n$$

该定义的直观意义是:通知声明部分的动态语义就是通知的复合。

4.4.3　命令的语义

在给定类声明和方面声明的情况下,称 Σ_{decl} 为声明的对象空间,其表示在声明部分已声明的所有对象的集合。序对 (Ω,Σ_{decl}) 称为程序上下文,用 Ξ_{decl} 表示。命令的语义主要是对命令执行后状态的改变进行描述。本语义采用动态编织进行描述,即在程序执行过程中,对连接点进行匹配并调用相应的通知。因此,本语义首先判断命令对应的连接点是否与通知匹配,如果匹配,则再根据通知的类型将通知体插入到命令的相应位置。下面对命令的语义进行定义。

定义 16　假设 c 为 AO 程序的命令,其语义定义如下:

$$[[c]]=_{df} \begin{cases} [[body(before(p))]][before(p).ADV/CJP(c).o];spec(c);[[body(after(p))]][after(p).ADV/CJP(c).o], \\ \qquad \text{if } p\in Pointcut \text{ s.t. } around(p)=o\wedge match(p.pcd,CJP(c))=p.PV[CJP(c).o] \\ [[body(before(p))]][before(p).ADV/CJP(c).o];[[body(around(p))]][around(p).ADV/CJP(c).o]; \\ \qquad [[body(after(p))]][after(p).ADV/CJP(c).o], \\ \qquad \text{if } p\in Pointcut \text{ s.t. } around(p)\neq o\wedge match(p.pcd,CJP(c))=p.PV[CJP(c).o] \\ [[body(before(p))]];[[body(aroun(p))]];[[body(after(p))]], \\ \qquad \text{if } p\in Pointcut \text{ s.t. } around(p)\neq o\wedge match(p.pcd,CJP(c))=1 \\ [[body(before(p))]];spec(c);[[body(after(p))]], \\ \qquad \text{if } p\in Pointcut \text{ s.t. } around(p)=o\wedge match(p.pcd,CJP(c))=1 \\ spec(c), \quad \text{otherwise} \end{cases}$$

其中,$spec(c)$ 为命令在面向对象中的语义,其定义见文献[10],$before(p)$ 表示切点 p 绑定的 $before$ 通知,$after(p)$ 表示切点 p 绑定的 $after$ 通知,$around(p)$ 表示切点 p 绑定的 $around$ 通知。

定理 2(命令语义保持定理)　当无方面声明,即 $adecls=\varepsilon$ 时,$[[c]]=spec(c)$。

证明:根据定义 16,此定理显然成立。

4.4.4　程序的动态语义

面向方面程序的动态语义是描述程序的行为,上文已定义声明部分[21-22]和单条命令的语义,本节将应用它们来定义程序的语义,其形式化定义如下:

$$[[cdecls\cdot adecls\cdot Main]]$$
$$=_{df}\exists\Omega,\Omega',internalvar,internalvar'\cdot[[cdecls]];[[adecls]];init;[[main]] \tag{3}$$

其中:

1)$\Omega=\{pricname,pubcname,superclass,pri,prot,pub,op,ASNAME,superaspect,ANAME,PNAME,Pointcut,Advice\}$。

2)$internalvar$:表示程序内部变量(程序中不在主方法中声明的变量)的集合。

3)$[[cdecls]]$:表示类声明部分的语义,见文献[10]。

4)$[[adecls]]$:表示方面声明部分的语义,见文献[21-22]。

5)$init$:表示程序的初始状态,其定义如下:$\mathbb{D}(cdecls\cdot adecls)\Rightarrow\{x,visibleattr,\Pi\};true\vdash(visibleattr'=\Phi)\wedge(\Pi'=\Phi)\wedge\bigwedge\limits_{x\in var}((\overline{x'}=\langle\rangle)\wedge(TypeSeq(x)'=\langle\rangle))$,其中 var 为局部

变量和外部变量(主方法中声明的变量)的集合,$visibleattr$ 表示程序执行过程中对命令可见的属性集,Π 表示已创建的对象的集合[10],$\overline{x'}$ 表示变量 x 的值序列,$TypeSeq(x)$ 表示变量 x 的类型序列。

6)$[[main]]$:表示主方法的语义,其定义见文献[9]。

定理 3(AO 程序动态语义保持定理) 当无方面声明,即 $adecls=\varepsilon$ 时,$[[cdecls \cdot adecls \cdot Main]] = [[cdecls \cdot Main]]$。

证明:$[[cdecls \cdot adecls \cdot Main]]$

$= \exists \Omega, \Omega', internalvar, internalvar' \cdot [[cdecls]]; [[\varepsilon]]; init; [[main]]$

$= \exists \Omega, \Omega', internalvar, internalvar' \cdot [[cdecls]]; init; [[main]]$

$= [[cdecls \cdot Main]]$

该定理表明,本节所定义的动态语义是面向对象程序的动态语义的扩充,即面向对象程序的动态语义是面向方面程序的动态语义的特殊情形。

5 示例

图 1 给出的示例程序包括一个类和两个方面的声明,其中类 Point 的声明记作 delPoint,方面 A 的声明记作 delA,方面 B 的声明记作 delB,类声明和方面声明分别是合适定义的[21-22],其对应的动态语义的计算过程如下。

步骤 1 先用本文定义的类 AspectJ 语言对示例程序进行描述:

```
class Point{
    int x;
    method:
    public int getX(Φ;int rx){rx:=x;}
    public void setX(int y;Φ){
    this.x=y;}
    public void add(int y;Φ){
    this.x=this.x+y;}
};
aspect A {
    pointcut p(Point d):call(void Point.add(int))&&target(d)
after adv1(Point e):p(e){
  e.getX(Φ;z));
  System.out.println(z);}
};
aspect B{
    pointcut p(Point d):call(void Point.add(int))&&target(d)
    after adv2(Point e):p(e){
      e.getX(Φ;z)
    System.out.println(z);}
}
class Main{
    public static void main(String[] args)
    {
    c:=Point.new()
      c.setX(100;Φ);
      c.add(5;Φ);
    }
}
```

步骤 2 计算$[[cdecls]]$。根据文献[10]的定义可得:

$$[[cdecls]]=true \vdash R1(\Omega,\Omega') \tag{4}$$

其中,$R(\Omega,\Omega')$表示 Ω_1 与 Ω_1'关系的谓词。

步骤 3 计算 $Pointcut$。根据定义 9 可得:

$p_1 = \langle A, p, Point, d, call(\text{void } Point.add(int)) \&\& target(d)\rangle$

$p_2 = \langle B, p, Point, d, call(\text{void } Point.add(int)) \&\& target(d)\rangle$

因此,$Pointcut=\{p_1,p_2\}$。

步骤 4 计算$[[adecls]]$。根据定义 10 可得:

$$adv_1 = \langle A, adv_1, after, Point, e, p_1, e.getX(\Phi;z)); System.out.printl n(z)\rangle \tag{5}$$

$$adv_2 = \langle B, adv_2, after, Point, d, p_2, e.getX(\Phi;z)); System.out.printl n(z)\rangle \tag{6}$$

因此:

$$Advice' = Advice \cup \{adv_1, adv_2\} \tag{7}$$

因为 $adv_1.advicetype=adv_1.advicetype=after$ 且 $adv_1.p_1.pcd=adv_2.p_2.pcd$ 且 adv_1 比 adv_2 先被定义,所以根据定义 14 可得:

$adv_1 \boxplus adv_2=\{Advice\}:\mathbb{D}(adv_1)\wedge\mathbb{D}(adv_2)\vdash Advice'=Advice-\{adv_1,adv_2\}\cup\langle adv_1.A, adv_1.ad, adv_1.advicetype, adv_1.ADVT, adv_1.ADV, adv_1.pc_1, adv_2.c_2[ADV_1/ADV_2]; adv_1.c_1\rangle$

根据式(5)—式(7)可得:

$adv_1 \boxplus adv_2=\{Advice\}:true \vdash Advice'=Advice\cup\langle A, adv_1, after, Point, e, p_1, e.getX(\Phi;z)); System.out.println(z); e.getX(\Phi;z)); System.out.println(z)\rangle$

为了叙述方便,令 $adv_3=\langle A, adv_1, after, Point, e, p_1, e.getX(\Phi;z)); System.out.println(z); e.getX(\Phi;z)); System.out.println(z)\rangle$,则有:

$adv_1 \boxplus adv_2=\{Advice\}:true \vdash Advice'=Advice\cup\{adv_3\}$。因此,根据定义 15 可得:

$$[[addecls]]=adv_1 \boxplus adv_2 = \{Advice\}: true \vdash Advice' = Advice \cup \{adv_3\}$$

根据文献[22]可得:

$$[[adecls]] = true \vdash R1(\Omega, \Omega'); \{Advice\}: true \vdash Advice'=Advice\cup\{adv_3\}$$

根据式(2)、定理 1、规则 1—规则 3,并将 Ω 中无关的变量隐藏,得到:

$$[[adecls]]=true \vdash Advice'=Advice\cup\{adv_3\} \tag{8}$$

步骤 5 计算$[[c:=Point.new()]]$。根据文献[9]可得:

$CJP(c:=Point.new())=_{df}\{\langle pcall,c,new,\varepsilon,\varepsilon\rangle,\langle pexecution,c,new,\varepsilon,\varepsilon\rangle,\langle finit,c,new,\varepsilon,\varepsilon\rangle,\langle fpreinit,c,new,\varepsilon,\varepsilon\rangle\}$

因为 $dtype(c)=Point, new\neq add$,所以根据定义 12 可得:

$match(call(\text{void } Point.add(int)),\langle pcall,c,new,\varepsilon,\varepsilon\rangle)=0$

$match(call(\text{void } Point.add(int)),\langle pexecution,c,new,\varepsilon,\varepsilon\rangle)=0$

$match(call(\text{void } Point.add(int)), \langle finit, c, new, \varepsilon, \varepsilon\rangle) = 0$

$match(call(\text{void } Point.add(int)), \langle fpreinit, c, new, \varepsilon, \varepsilon\rangle) = 0$

$match(call(\text{void } Point.add(int)), CJP(c := Point.new())) = 0$

$match(call(\text{void } Point.add(int)) \&\& target(d), CJP(c := Point.New()))$

$= match(call(\text{void } Point.add(int)), CJP(c := Point.new()))(match(target(d), CJP(c := Point.new())))$

$= 0$ (9)

根据式(9)、文献[10]和定义 16 可得:

$[[c := Point.new()]]$

$= spec(c := Point.new())$

$= \{c, \Pi(Point)\}: true \vdash (\Pi(Point)' = (\Pi(Point) \uplus \{\langle r, Point, \{\langle x \mapsto 0\rangle\}\rangle) \wedge (\overline{c} = \langle r, Point\rangle \cdot tail(\overline{c})) \wedge (TypeSeq(c)' = \langle Point\rangle \cdot tail(TypeSeq(c)))$ (10)

步骤 6　计算$[[c.setX(100, \Phi)]]$。使用步骤 5 的方法可得:

$[[c.setX(100, \Phi)]]$

$= \{\Pi(Point)\}: true \vdash ((\Pi(Point)' = \Pi(Point) \uplus \{\langle c, Point, \{x \mapsto 100\}\rangle\})$ (11)

步骤 7　计算$[[c.add(5, \Phi)]]$。使用步骤 5 的方法可得:

$[[c.add(5, \Phi)]]$

$= \{\Pi(Point), z\}: true \vdash (\Pi(Point)' = \Pi(Point) \uplus \{\langle r, Point, \{x \mapsto x+5\}\rangle\}) \wedge (\overline{z'} = \langle x\rangle \cdot \langle 0\rangle) \wedge (TypeSeq(z)' = \langle int\rangle\})$ (12)

步骤 8　计算$[[cdecls \cdot adecls \cdot Main]]$。根据式(3)可得:

$[[cdecls \cdot adecls \cdot Main]]$

$= \exists \Omega, \Omega', internalvar, internalvar'[[cdecls]]; [[adecls]]; init; [[main]]$

根据文献[10]可得:

$[[cdecls \cdot adecls \cdot Main]]$

$= \exists \Omega, \Omega', internalvar, internalvar'[[cdecls]]; [[adecls]]; init; [[c := Point.new()]]; [[c.setX(100, \Phi)]]; [[c.add(5, \Phi)]]$

根据式(4)、式(8)、式(10)－式(12)及文献[10]可得:

上式 $= true \vdash R1(\Omega, \Omega'); true \vdash Advice' = Advice \cup \{adv_3\}; \{x, visibleattr, \Pi\}: true \vdash (visibleattr' = \Phi) \wedge (\Pi' = \Phi) \wedge (= \langle\rangle)(TypeSeq(c)' = \langle\rangle)(= \langle\rangle)(TypeSeq(e)' = \langle\rangle); (\Pi(Point)' = \{\langle r, Point, \{x \mapsto 105\}\rangle\}) \wedge (\overline{c'} = \langle r, Point\rangle) \wedge (TypeSeq(c)' = \langle Point\rangle)) \wedge (\overline{z'} = \langle 105\rangle \cdot \langle 0\rangle)((TypeSeq(z)' = \langle int\rangle)$

根据定理 1,规则 1－规则 3,并将 Ω 中无关的变量隐藏,可得:

$[[cdecls \cdot adecls \cdot Main]]$

$= true \vdash (visibleattr' = \Phi) \wedge (\Pi(Point)' = \{\langle r, Point, \{x \mapsto 105\}\rangle\}) \wedge (\overline{c'} = \langle r, Point\rangle) \wedge (TypeSeq(c)' = \langle Point\rangle)) \wedge (\overline{z'} = \langle 105\rangle \wedge \langle 0\rangle) \wedge (TypeSeq(z)' = \langle int\rangle)$

此语义表示该程序创建一个类 *Point* 的对象 c,其属性 x 的值为 105,且通过 z 返回 x 的值。此语义与程序的实际运行效果一致。

结束语　本文在文献[20-22]的基础上定义了 AOP 的动态语义,该语义用于对面向方面程序的行为进行形式化描述,以对程序的正确性进行验证,从而保证软件开发和再工程过程的正确性。

本文所定义的语义是源语言级别的,不需要转换就能直接表示面向方面程序的行为,从而容易被熟悉 ASPECTJ 的开发人员接受,为 AO 程序的形式化研究奠定了基础。

参 考 文 献

[1] KICZALES G, LAMPING J, MENDHEKAR A, et al. Aspect oriented programming[C]// European Conference on Object Oriented Programming. Springer Berlin Heidelberg, 1997: 220-242.

[2] 王砚霖,王世耆. 面向方面编程和 AspectJ[OL/EB]. [2016-03-01]. http://www.creativepioneer.com/paper/AO P_and _AspectJ.pdf.

[3] KICZALES G, HILSDALE E, HUGUNIN J, et al. An overview of AspectJ[C]// European Conference on Object-Oriented Programming. Springer Berlin Heidelberg, 2001: 327-335.

[4] SPINCZYK O, GAL A, SCHRÖDER-PREIKSCHAT W. AspectC++: an aspect-oriented extension to the C++ programming language[C]// Proceedings of the Fortieth International Conference on Tools Pacific: Objects for Internet, Mobile and Embedded Applications. Australian Computer Society, Inc., 2002: 53-60.

[5] BRYANT A, FELDT R. Asepct R-Simple aspect-oriented programming in Ruby[OL]. http://aspecter-for.sourceforgenet.

[6] BONÉR J. What are the key issues for commercial AOP use: how does AspectWerkz address them? [C]// Proceedings of the 3rd International Conference on Aspect-oriented Software Development. ACM, 2004: 5-6.

[7] HIRSCHFELD R. AspectS-Aspect-oriented programming with squeak [M]// Objects, Components, Architectures, Services, and Applications for a Networked Word. 2002: 216-232.

[8] JBoss AOP homepage[OL/EB]. [2016-03-01]. http:// www.jboss.org/jbossaop.

[9] OSSHER H, TARR P. Hyper/J: multi-dimensional separation of concerns for Java[C]// Proceedings of the 22nd International Conference on Software Engineering. ACM, 2000: 734-737.

[10] JIFENG H, LI X, LIU Z. rCOS: A refinement calculus of object systems[J]. Theoretical Computer Science, 2006, 365(1): 109-142.

[11] WAND M, KICZALES G, DUTCHYN C. A semantics for advice and dynamic join points in aspect-or iented programming [J]. Acm Transactions on Programming Languages and Systems, 2004, 26(5): 890-910.

(下转第 207 页)

classification[J]. Information Processing & Management，2014，50(1)：104-112.

[7] D'ASPREMONT A. Predicting abnormal returns from news using text classification[J]. Quantitative Finance，2015，15(6)：999-1012.

[8] SHANG C，LI M，FENG S，et al. Feature selection via maximizing global information gain for text classification[J]. Knowledge-Based Systems，2013，54(4)：298-309.

[9] KANAAN G，AL-SHALABI R，GHWANMEH S，et al. A comparison of text-classification techniques applied to Arabic text [J]. Journal of the American Society for Information Science & Technology，2014，60(9)：1836-1844.

[10] KHORSHEED M S. Comparative evaluation of text classification techniques using a large diverse Arabic dataset[J]. Language Resources & Evaluation，2013，47(2)：513-538.

[11] ABUERRUB A. Arabic Text Classification Algorithm using TFIDF and Chi Square Measurements[J]. International Journal of Computer Applications，2014，93(6)：40-45.

[12] HU J，YAO Y. Research on the Application of an Improved TFIDF Algorithm in Text Classification[J]. Journal of Convergence Information Technology，2013，8(7)：639-646.

[13] GHAG K，SHAH K. SentiTFIDF-Sentiment Classification using Relative Term Frequency Inverse Document Frequency[J]. International Journal of Advanced Computer Science & Applications，2014，5(2)：36-43.

[14] BILAL M，ISRAR H，SHAHID M，et al. Sentiment classification of Roman-Urdu opinions using Navie Baysian，Decision Tree and KNN classification techniques[J]. Journal of King Saud University-Computer and Information Sciences，2016，28(3)：330-344.

[15] CHEN R，CHEN F，SUN Y. Research on Automatic Text Classification Algorithm Based on ITF-IDF and KNN[J]. Applied Mechanics & Materials，2015，713-715：1830-1834.

[16] FENG G，WANG H，SUN T，et al. A Term Frequency Based Weighting Scheme Using Naïve Bayes for Text Classification [J]. Journal of Computational & Theoretical Nanoscience，2016，13(1)：319-326.

[17] GONG W，CAI Z. Differential evolution with ranking-based mutation operators[J]. IEEE Transactions on Cybernetics，2013，43(6)：2066-2081.

[18] YANG M，GU J. Study and Apply of Chinese Bibliographies Automatic Classification Based on Support Vector Machine[J]. Library and Information Service，2012，56(9)：114-119. (in Chinese)
杨敏，谷俊. 基于 SVM 的中文书目自动分类及应用研究[J]. 图书情报工作，2012，56(9)：114-119.

[19] PAULINAS M. A survey of genetic algorithms applications for image enhancement and segmentation[J]. Information Technology & Control，2015，36(3)：278-284.

[20] JIN X R，QI J D，WANG L C，et al. Approach of classification mapping between international patent--classification and chinese library classification based on machine learning[J]. Journal of Computer Applications，2011，31(7)：1781-1784. (in Chinese)
靳雪茹，齐建东，王立臣，等. 基于机器学习的类目映射方法——国际专利分类法与中国图书馆分类法[J]. 计算机应用，2011，31(7)：1781-1784.

[21] YANG B，HAN Q W，LEI M，et al. Short Text Clssification Algorithm Based on Improved TF-IDF Weight [J]. Journal of Chongqing University of Technology (Natural Science)，2016，30(12)：103-113. (in Chinese)
杨彬，韩庆文，雷敏，等. 基于改进 TF-IDF 权重的短文本分类算法[J]. 重庆理工大学学报(自然科学)，2016，30(12)：103-113.

(上接第 185 页)

[12] JAGADEESAN R，JEFFREY A，RIELY J. A calculus of untyped aspect-oriented programs[C]// European Conference on Object-Oriented Programming. Springer Berlin Heidelberg，2003：54-73.

[13] LÄMMEL R. A semantical approach to method-call inte rception[C]// Proceedings of the 1st International Conference on Aspect-oriented Software Development. ACM，2002：41-55.

[14] WALKER D，ZDANCEWIC S，LIGATTI J. A theory of aspects [J]. Acm Sigplan Notices，2003，38(9)：127-139.

[15] TUCKER D B，KRISHNAMURTHI S. Pointcuts and advice in higher-order languages[C]// Proceedings of the 2nd International Conference on Aspect-oriented Software Development. ACM，2003：158-167.

[16] MASUHARA H，KICZALES G. Modeling crosscutting in aspect-oriented mechanisms[C]// European Conference on Object-Oriented Programming. Springer Berlin Heidelberg，2003：2-28.

[17] TABAREAU N. Aspect Oriented Programming：a language for 2-categories[C]// Proceedings of the 10th International Workshop on Foundations of Aspect-oriented Languages. ACM，2011：13-17.

[18] MOLDEREZ T，JANSSENS D. Modular Reasoning in Aspect-Oriented Languages from a Substitution Perspective[C]// Transactions on Aspect-Oriented Software Development XII. Springer Berlin Heidelberg，2015：3-59.

[19] ZHANG Q，KHEDRI R. On the weaving process of aspect-oriented product family algebra[J]. Journal of Logical and Algebraic Methods in Programming，2016，85(1)：146-172.

[20] GANG X，BO Y，MINGYI Z. A Semantics of Poincuts in AspectJ[J]. IERI Procedia，2013，4：323-330.

[21] XIE G，ZHANG M Y，YANG B. A Static Sema Ntic For Aspectj [J]. Journal of Computational Information Systems，2012，8(16)：6951-6962.

[22] XIE G，WEI L，WU X. static semantics of aspect-oriented programming[J]. Computer Science，2017，44(9)：184-189. (in Chinese)
谢刚，韦立，吴祥. 面向方面程序的静态语义研究[J]. 计算机科学，2017，44(9)：184-189.

[23] 陆钟万. 面向计算机科学中的数理逻辑(第 2 版)[M]. 北京：科学出版社，2002：117-118.

[24] HOARE A R C，HE J. Unifying theories of program ming[M]. Englewood Cliffs：Prentice Hall，1998.