增量静态语义分析的一个对象模型*

徐智晨 钱家骅

(复旦大学计算机科学系,上海 200433)

摘要 本文给出了一个对象模型用以描述类PASCAL程序设计语言的静态语义(语义对象,语义依赖).讨论了基于该模型的增量静态语义分析的过程,动作,并发机制与实现手段.在文章的最后部分对模型扩充讨论了在多用户,分布式环境中的增量语义分析.文中的模型可稍加扩充以适应一般的高级程序设计语言.

关键词 增量静态语义分析,面向对象,类PASCAL语言,属性文法,多用户、分布式环境.

程序的增量静态语义(文中"语义"如不作特别说明均视为静态语义)分析仅计值程序修改所波及的最小范围,速度大大快于对应的非增量型实现.该技术的采用可在软件(程序)开发过程中及早地将静态语义错提示给开发者.在编码阶段的早期杜绝致命性的语义错[5].增量的静态语义分析还可以为高级的排错工具增量地提供信息以支持及时地排错[5].特别的,在多用户、分布式环境中使用可在多个开发者之间及时传递信息,及早地报告多个开发者所开发对象之间语义的不一致,协调多个开发者之间的合作.

所谓程序语言的静态语义是指该种语言程序都必须满足的上下文相关的约束.静态语义分析则旨在具体的程序中找出这些约束并检查其满足程度.具体而言,程序静态语义分析的主要任务可归结为3个方面:语义检查,语义推理,为程序的动态语义分析提供信息支持.

语义检查则为检查程序对上述上下文相关约束的满足性.语义推理则是找出程序上下文中被隐含的或不完整的约束关系,(如带多态类型语言中的类型推理).为程序的动态语义分析提供信息支持则是指那些程序执行中会用到的静态可计算信息的计算(如静态的地址分配等).

本文基于实用高效的角度,描述了类PASCAL语言程序增量静态语义分析的一个语义模型.该模型虽针对类Pascal语言,但可稍加扩充以适应一般的高级程序设计语言.

1 类PASCAL程序语言语义的对象模型

类PASCAL程序语言的最主要的特点是分程序结构、块结构的最近作用域规则以及丰

* 本文1991-12-07收到,1992-07-06定稿

作者徐智晨,31岁,博士生,主要研究领域为领域模型化,需求获取,知识表示及推理,CASE工具.钱家骅,享年64岁,生前曾支持多项"七五"、"八五"和"八六三"高技术项目,获得多项部、市委级科技进步奖.

本文通讯联系人:金立群,上海200433,复旦大学软件工程课题组

富的数据类型.对于类 PASCAL 程序语言,静态语义分析的主要任务可进一步具体地分为:数据流分析、类型检查(推理)、作用域约束和静态地址分配.数据流分析包括类型、数据、函数(过程)的定义,说明与引用关系的分析.类型检查则为检查类型的相容性.在增量环境中由于允许数据的先引用后定义,类型推理机制则可以根据数据在程序中的上下文推出未经说明定义数据可能的类型.作用域约束与静态地址分配均是为程序动态语义分析提供信息支持.

本文的对象模型由语义对象类、语义对象、语义对象类(或语义对象)之间的联系和层次型环境结构组成.其中,语义对象类又划分为类型语义对象类,数据语义对象类和数据应用语义对象类.这些对象类之间的关系服从图 1 所示的继承层次结构.语义对象则为语义对象类的实例.

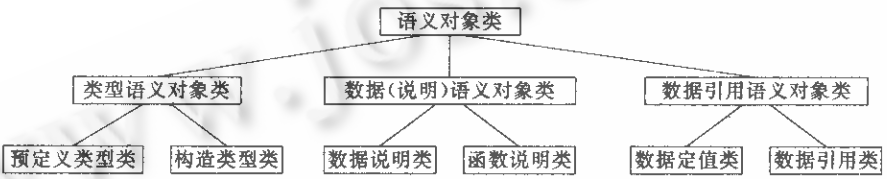


图1

语义对象类之间的联系(本文中以后简称为语义联系)是类型定义、数据定义、数据说明引用关系、数据定值—引用关系,类型匹配关系的抽象.如果,我们记类型为 T,数据说明为 D,数据定值为 Dd,数据引用为 Du,那么,上面提到的语义联系可表示为表 1 所示的映象关系.

表 1

类型定义	$T^* \rightarrow T$
数据定义	$T \rightarrow D$
数据说明引用	$D \rightarrow Dd \mid Du$
数据定值引用	$Dd \rightarrow Du$
类型匹配	$T^* \rightarrow T$

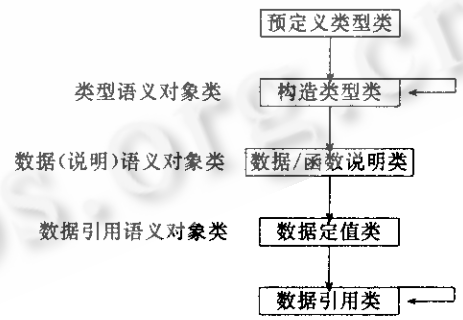


图2

文中对象模型中的语义联系则为表 1,所有映象关系的综合,我们统一称之为"定义—引用"联系.这样,类 PASCAL 程序语言静态语义的依赖关系可抽象成如图 2 所示的多级"定义—引用"模型.

图 2 给出了程序语义对象之间依赖关系的一个概念模型.其具体的依赖关系由程序中的产生式实例(包括非局部产生式)给出.这种"定义—引用"联系在文中的模型中实际由语义对象中属性所定义的映象给出.表 2 列出了模型中各语义对象应包含的属性.

表2

```
类型语义对象类:
TYPE_LENGTH;
RELATIVE_OFFSET;————————————/*静态地址信息*/
TYPE_CONSTRUCT_INFO;————————————/*类型构造信息*/
DEFINED_BY;SET(Semantic_Object);————————/*引用类型集合*/
DEFINES;SET(Semantic_Object);————————/*定义类型集合*/
……
数据(说明)语义对象类:
RELATIVE_OFFSET;————————————/*静态地址信息*/
DATA_CONSTRUCT_INFO;————————————/*数据构造信息*/
DEFINED_BY;SET(Semantic_Object);————————/*引用类型集合*/
DEFINES;SET(Semantic_Object);————————/*定义类型集合*/
……
数据引用语义对象类;
SITE_INFO;————————————/*对象所在场地信息*/
DEFINED_BY;SET(Semantic_Object);————————————/*数据定义信息*/
VALUE_DEFINED_BY;————————/*数据定值信息*/
VALUE_REFERENCED_BY;————————/*数据引用信息*/
```

类PASCAL程序语言支持分程序结构.服从块结构的最近静态作用域规则.为了便于描述类PASCAL程序语言最近静态作用域规则这类特殊的语义依赖,本文在提出的语义模型中引入了层次型的环境结构(图3a),环境在文中的语义模型中为语义对象实例的属主结构.任何一个语义对象实例都必须且只能落在一个环境中.环境结构还可由图3b的仰视图形象地表示.图3中的层次结构具有如此的意义.

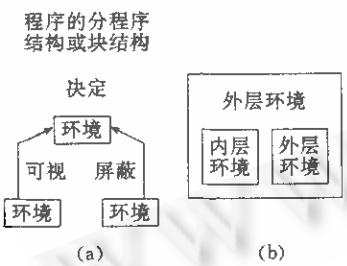


图3

```
main( )
{
int X,Y
if shadow>1
{int X;
 X++;Y++;
 }
else
{int Y;
 X++;Y++;
 }
}
```

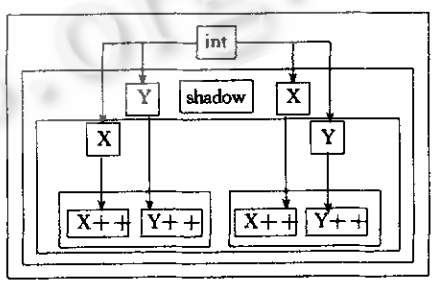


图4

下层环境可视其任何祖先环境中的内容;下层环境中说明的数据,函数可屏蔽其祖先环境中定义(说明)的同名数据/同基调函数.

至此,我们已介绍了将作为本文增量静态语义分析过程讨论基础的类PASCAL程序语言的语义描述模型.它由语义对象类,语义对象,语义联系以及层次型环境结构几个核心概念组成.在结束本文之前我们将给出一个实例,帮助读者加深对文中语义模型的理解.图4左部的程序段的语义由图4右部的模型实例给出.

2　基于对象模型的增量静态语义分析

增量的语义分析对由程序修改引进的语义变动进行增量的传播,或使原来不一致的语义达到一致,或引起新的语义不一致提示开发者修正.增量性在这里体现在仅对因程序修改所波及的最小范围进行分析.对于文中提出的对象模型.语义变动的传播过程可抽象成:

多级"定义—引用"对象模型实例的构建与调整过程;

在多级"定交—引用"对象模型实例中,从高层向低层的语义的传播过程;(定义点对用点的影响)

在多级"定义—引用"对象模型实例中,自低层向高层的语义传播过程;(引用点对定义点的影响)

在增量的环境中应允许先引用后定义的情况存在.本文提出"影子"对象的概念来描述那些已被使用但未实际被定义或说明的语义对象.如图4,C语言程序段中的变量"shadow"."影子"对象一略被放置在包含该"影子"的最小块或分程序结构所定义的"环境"中.这样做是因为我们当前无从得知它将在哪个"环境"中实际被定义.这一"影子"对象的存在保证了"伪"的语义一致性.这种可因将来对程序进行添加而消除的语义不一致,我们称之为可容忍性语义错[5],这个"影子"可因将来对相应的语法单元的实际说明或定义得到归结从而达到真正的语义一致性.

基于文中提出的对象模型.增量静态语义分析涉及二类算法,即增加一个语义对象对应的分析算法与删除一个语义对象对应的分析算法.本文分别称之为"增算法"与"减算法"它分别由算法1和算法2给出.

算法1　增算法:INSERT(NewlyADD)

```
/*———— DEFINES 为所有语义对象共有的属性,
—*————它用于存放由该对象定义的其它对象;
—*————函数 ENV 返回一语义对象所处的环境.
—*/
设 NewlyADD 为新添对象;
IF(在当前环境中已存在与 NewlyADD 同名同类的对象,
  且,NewlyADD 类对象不允许在同一环境中重复出现
  )
  ERROR({DUBLICATED OBJECT DEFINITION});
IF(在祖先环境中存在与 NewlyADD 同名的同类对象,设为 OUT)
  {
   OUT.DEFINES=OUT.DEFINES−{X|X∈OUT.DEFINES∧X∈ENV(NewlyADD)};
   NewlyADD.DEFINES={x|x∈OUT.DEFINES∧X∈ENV(NewlyADD)};
  }
ELSE IF(在 ENV(NewlyADD)中存在与 NewlyADD 同名同类的影子,设为 SHADOW)
    {
     NewlyADD.DEFINES=SHADOW.DEFINES;
     向 NewlyADD 所定义的对象传播语义信息.
     DELETE(SHADOW);
```

```
IF (从当前环境起存在这样一个对象 OD 使得:存在 X∈OD.DEFINES,
   且,X 与其 OD 同名同类
   )
   {/* 针对数据函数引用点 */
   OD.DEFINES=OD.DEFINES ∪ { NewlyADD };
   计算有关 NewlyADD 的语义信息.
   }
ELSE
      {
      在 ENV(NewlyADD)中增加一个影子 SHADOW;
      SHADOW.DEFINES={ NewlyADD };
      /*SHADOW 为 NewlyADD 对应的定义类*/
      }
```

算法 2 减算法

```
/*defines,Env 同算法 1 */
设 ToBeDELETED 为将被删除的对象
if (ToBeDELETED.defines 非空)
{
 if (从当前环境起能找到与 ToBeDELETED 同名同类的对象设为 OutO)
    OutO.defines=OutO.defines ∪ ToBeDELETED.defines;
 else{
      生成一个与 ToBeDELETED 同名同类的影子对象 shadowO;
      shadowO.defines=ToBeDELETED.defines;
      }
 }
if(从当前环境 Env(ToBeDELETED)里层向外相找这样一个对象 Od,
   ToBeDELETED∈Od.defines
   )
   Od.defines=Od.defines−(ToBeDELETED);
在 Env(ToBeDELETED)中删去 ToBeDELETED;
```

对应本文提出的对象模型,增量语义分析过程被抽象成为环境结构上多级"定义一引用"链的构建与维护以及语义信息沿多级"定义一引用"链语义的向前,向后传播过程.算法1和算法2忽略了许多语义分析的细节,算法框架可看成是多级"定义一引用"链的构建与调整算法.全部的语义分析动作事实上可抽象为:构造动作,删除动作,更新动作.

为了更好地开发增量静态语义分析的并发性,保证语义模型实例的完整性,上面谈到的语义动作又可以进一步置于一些活动中(Activity).每个语义动作的执行只能在某一个活动框架中执行.这些活动分为:

不受保护的:它在对某一语义对象实施操作时不具排它性.

受保护的:它在对某一语义对象实施操作时具有排它性,但不必保证其执行的原子性.

事务:它的执行既是排它的又必须保证原子性.

文中提出的模型具有非常简单的结构.较高的抽象层次,能够较全面地表达高级程序设计语言的静态语义信息.基于该模型的静态语义分析过程实际上是多级"定义一引用"链的

构建调整过程以及语义信息在多级"定义一引用"链上传播过程.整个分析算法具有较高的抽象层次,便于开发具有高度并行度的具体算法.本文中的算法除可直接支持面向对象的实现方案之外,还可以有多种其它实现方案供选择.属性文法语义描述特别适合文中增量静态语义分析方式的实现.采用属性文法,程序段表示成饰以属性的抽象语法树.语义对象和环境构成用聚合属性描述.对于程序语义的属性文法表示,语义分析过程或采用动作例程法、或采用增量属性计算值技术.

3　多用户,分布式环境中的静态语义分析

为了使基于语言的程序设计环境或面向语言的程序编码排错工具能应用于大规模的程序设计,支持多用户的协调开发,研究多用户,分布式环境中的增量静态语义分析技术显得非常重要.

本文下面的讨论基于这样一个假设:一个程序由若干个模块组成.每个开发人员管理整数个模块,一个模块不能被两个以上开发者同时修改.

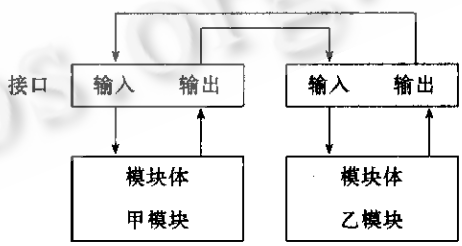
多用户,分布式环境中的增量静态语义分析最关键的问题是如何将对某一模块修改所引起的语义变化传播给所有引用该模式的其它模块.这在多用户环境下成为问题是因为接受语义传播的那个模块有可能正处在被编辑的状态.语义传播不能象在单用户环境中那样立即进行.

3.1　多用户非分布式环境中的增量静态语义分析

为了能在多个模块间传播语义变动.我们在文中对象模型中增加了接口对象的概念.接口对象包含一个输入部分与一个输出部分,分别表示本模块对它模块的引用和本模块可能对它模块的影响.模块接口与模块体之间的语义依赖以及各模块之间的语义联系如图 5 所示.

图中,模块接口输出部分的语义依赖于模块体语义,模块接口输入部分的语义向模块体传播.一模块的接口输出部分决定所有引用它的其它模块的接口的输入部分.

当语义箭头方向传播时,或使原来不一致的语义达到一致,或引起新的语义不一致,报语义错以提示开发者修正.



图中空箭头表示语义传播(决定)方向

图5

无论是模块接口对象的输入部分还是输出部分都由类型定义、数据定义,函数定义组合而成.接口对象是复合对象.

此时语义分析过程是这样的:当一个模块被修改时首先进行模块内语义分析.如果,这一修改也引起了该模块接口语义的变动.则将这一变动向所有引用该模块的其它模块传播.传播首先影响其它模块接口的输入部分,由接口输入部分继续向模块体传播,或使语义达到一致,或引起新的语义不一致.

前面我们已提到,在多用户情况下可能在实际语义传播时接受模块正处在编辑状态.如

图3中甲模块被修改完毕并准备向乙模块传播语义信息时乙模块正在被编辑.为了解决这个问题,我们给个模块加了一个阀,阀可以是打开的也可以是关闭的.这有在阀处打开状态时才能对该模块实施语义传播.

这里有几点值得注意:(1)文中模块的概念是广义的.它可以是一个通常意义下的模块也可以是整数个通常意义下的模块.(2)模块上的阀门应该是可以受开发者控制,这一传播是立即的,也可以是延迟的.Mecury[3]提出了修改模拟(change simulation)这一有趣的概念.意思是:程序员可对他主管的模块实施假设的修改,看一看这种修改会有什么样的影响.对于文中提出的模型,一种解决办法是,首先复制本模块和其它模块的接口信息各一份.不复制其它模块体是因为接口输入部分已抽取了模块对外引用的全部信息.在复制之后,对这些复制品进行修改和语义传播.

3.2 多用户分布式环境中的增量静态语义分析

分布式环境中增量静态语义分析中的新问题来自于程序与开发者都分布在网络的多个场地之上.为了讨论问题简单我们进一步对模型提出假设:模块要么整个在一个场地存在,要么整个不存在.如果网络上存在同一模块的多个拷贝的话,只有一个能处在写状态.其它拷贝在没有得到相应的更新前被认为是废弃的,且不参加语义传播.

在分布式环境中,语义信息的传播必须通过网络场地间发送邮包完成.

第一个问题是:如果一个模块连续被做了多次修改,它所在的场地因此向其它场地发送了一些邮包.而网络却打乱了这些邮包的时间顺序.

一种解决办法是,每个邮包在发送之前都打上对应于本次修改的时间标记.接受场地在收到多个同一场地同一模块发出的邮包时,仅取时间最新的那个邮包.这里不要求各个场地的时钟同步,因为只需对同一模块发出的邮包的时间标志作比较且一个模块只能在一个场地上被修改.

第二个问题是:如果网络在传送邮包时失败怎么办?

一个解决方案是一个场地在受到它场地的邮包后向发出场地发一个收执.发出场地在一定时间内收不到回执时,就按邮包原来的时间标志再发送一次.如发出场地又有了新的传送任务(还是由于对原来那个模块的修改)则发送场地不再继续发送那些传送失败的邮包以避免重复.

本节我们讨论了"多作户一分布式"环境中的增量静态语义分析问题.其中最关键的是:(1)每个模块对应一个接口语义对象存放该模块的输入/输出信息.(2)每个模块增加一个语义传播"阀门"以解决对模块实施语义传播动作与对模块实施编辑动作的冲突.(3)在分布式环境中通过场地间发送邮包传递语义信息.每个邮包带有一个时间标志以保证最新语义信息能被识别并实施传播,场地间通过发回执与不断发送丢失的邮包以消除因网络通迅失败造成的影响.

4 总 结

本文给出了一个对象模型用以描述类PASCAL程序语言的静态语义.文中的模型概念简单一致,易于理解且有较高的抽象层次,既能充分反应静态语义分析的关键问题,又方便实际实现.特别本文还基于该模型分析了"多用户分布式"环境中的增量静态语义分析.该模

型可稍加调整用于其它高级程序设计语言.本文作者基于该模型实现了国家"七五"攻关项目集成式软件式程支撑环境"青鸟"中面向语言的程序编码排错工具"CCD"中的增量静态语义分析[6].

参考文献

1 Knuth D E. Semantics of context—free languages. Mathematical Systems Theory, 1968:127—145.

2 Reps T, Teitelbaum T, Demers A. Incremental context—dependent analysis for language—based editors. ACM Trans. Programming Languages and Systems, 1983:449—478.

3 Gail E Kaiser, Simon M Kaplan, Josephine Micallef. Multiuser, distributed language—based environment. IEEE Software, 1987.

4 Lee P, Piecan U. A realistic compiler generator based on high—level semantics. Ergo Report 88—057. The University of Michigan, 1988.

5 徐智晨.面向对象的软件开发方法,与面向语言的程序设计环境.硕士学位论文,复旦大学,1990.

6 金立群等.面向语言的程序编码与排错工具CCD.第四届全国软件工程会议论文集.北京,1991.

AN OBJECT MODEL FOR INCREMENTAL STATIC SEMANTIC ANALYSIS

Xu Zhichen and Qian Jiahua

(*Department of Computer Science, Fudan University, Shanghai* 200433)

Abstract An object model is provided to model the static semantic of PASCL—like programming language in this paper. Based on this model the authors discussed the process, actions, parallel machanism and implementation alternatives of incremental static semantic analysis. In the last part of this paper they further adapted the model to make it suitable for multiuser, distributed environment. The model can be easily tuned for other high—level programming language.

Key words Incremental static semantic analysis, object orientation, PASCAL—like programming language, attribute grammar, multi—user, distributed environment.