限制的对象表示和计算*

奚建清 王能斌

(东南大学计算机系 南京 210018)

摘要 限制描述了对象间的一组复杂关系,现有OOPL中一般不支持它们的表示和计算.本文给出了一种限制的对象表示法,其思想是在引入限制的同时不引入额外的概念,保持对象模型和语言的简单性.本文还讨论了这种表示法引入的问题和解决方法.

关键词 限制,限制求解,对象,面向对象程序设计,程序设计.

1 为什么要引入限制

对象这一概念的直观意义就是引导人们首先以相对孤立、局部的角度去分析问题域中的事物,即先注重描述各个组成未来计算系统的对象或互相紧密相关的对象群的性质和行为,尔后才考虑对象之间的联系,从OOPL程序上看,对象被其名字、属性和方法以及产生过程清晰地确立了它在系统中的存在,而对象之间的关系则一般隐含在属性、方法等代码中,在传统的OOPL如Smalltalk和C++中,用户除了在方法设计和类设计过程中能控制对象之间的关系外,无其它手段来直接了解和控制对象之间的关系.

例如对一个并行四边形的架子,该架子可以活动(移动、变形),但必须保持平行四边形形状.如果把它的四条边作为对象,每条边的两个顶点坐标作为该线对象的属性,则在四边形变形时,应保证边对象顶点属性之间的关系,即相邻边的公共顶点坐标相等.这个关系可以有几种方式表示,一是边对象修改属性的方法负责检查参数值是否满足关系.显然,当有各种四边形时,如果该方法考虑各个四边形的具体情形,就会使该方法十分复杂.第二种方式是先为每个组成四边形的4个边对象设立一个特殊的边对象类,由该类定义的属性修改方法考虑到这4个边对象之间的关系.第三种方式是把四边形看成是一个对象,并且是由4个边对象复合而成.边对象的移动操作由四边形对象变形或移动操作控制,后一操作可以考虑到局部于该四边形对象的4个边属性之间的关系.这种方法需要设计者不断地抽象出新的类(如上述的四边形),有时这样做是十分勉强的(例如,两个相关对象并不在认识上组成一个复合对象的情形).另一存在问题是关系被"固化"在方法代码中,这不仅使这种关系隐含起来,而且任何对这种关系的修改就要涉及到对包含这个关系的方法的全部或部分的修改.例如,当设计平行四边形的一个子类正方形时,正方形的变形或移动操作就必须加强原

---

* 作者奚建清,1962年生,工程师,主要研究领域为OODB,OOPL,AI.王能斌,1929年生,教授,博士导师,主要研究领域为数据库和信息系统.

本文通讯联系人:奚建清,南京 210018,东南大学计算机系

本文1994-07-18收到,1994-11-21定稿

有关系,即保持相邻边之间的垂直关系,这需要重设计原来的变形或移动方法.而当子类为一个任意四边形,且各边长度可变时,必须打破原有的关系(如对边长度相等的关系).回避这种关系的放宽是先设计任意四边形,后设计平行四边形作为它的子类,即需要在设计类时进行计划(planning).第3个问题是由于关系的"固化",不能利用这些约束来求解一些未知量,例如据平行四边形中两条边的位置推出另两条边的位置.

由于以上原因,我们认为必须提供明显描述以及控制对象间关系的手段,这对于提高系统的表达能力、分散应用设计中的复杂控制流、方便用户编程和修改、增加程序的可理解性都是有很大的帮助.

2 已有的一些限制描述语言

在传统语言中,逻辑语言如Prolog、等式系统等均可描述变量间的关系,Prolog还可以直接描述关系之间的蕴涵关系.在面向对象系统中,已有一些系统把关系作为应用系统的主要构件,它们有DSM、thinglab和Contract等,下面对它们分别作一简单的介绍.

2.1 Contract结构[1]

Helm等人提出了一种叫Contract的结构,这种结构明显地描述了一组对象的行为限制.Contract的新颖之处在于它是可复合的,通过使用Contract Refinement和Contract Inclusion声明,一个新的Contract可以包括已有的Contract或细化一个已有的Contract,一个Contract抽象描述了一组参与者之间的动静态联系,包括对每个参与者的结构的要求,动作之间的互相牵制关系.通过对一个Contract中的参与者实例化(instantiation),即可使这个Constract规定的关系落实到一组对象上,这个过程称为class conformance,事实上Contract是用于产生类的,通过影射一个类定义的属性和方法到Contract中参与者的属性和方法,即可使这个contract的关系成立于该类的所有实例上.因此,一个Contract在很大程度上类似于一个类的抽象定义,不能用来动态地修改实例对象之间的限制关系.

2.2 Thinglab[2-4]

这个系统是建立在Smalltalk—80上的限制描述和求解系统,一组成员对象构成了一个thing,thing同时也包括了它的成员必经满足的限制,这些限制按重要性排成一个序,如按required、strongly prefered和prefered等排序,在求解时required限制是必须满足的,其它限制是尽可能地按序予以满足.每个限制包含了一个谓词P,一个偏差函数error和一组方法,这组方法中任何一个方法的运行都会使该限制得到满足.这比Contract前进了一步,即可让用户自定义满足限制的各种方法.限制可动态发生作用并被用来计值(即求解),每个限制的满足方法必须由用户给出.Thing是应用系统的主要构件,并在对象世界之外生成.

2.3 DSM[5,6]

DSM引入了关系,虽然Rumburg声明要把关系作为和类同样原始的机制,但关系仅描述(或保存)了一组对象之间特别是一个簇集(cluster)中的简单的模式(scheme),如DEFINE belongs—to Relation club * — * members:Personp定义一个club和Person之间的各对象的一个关系.这种关系不能用于动态控制对象的行为,一种用法是可以根据这种关系对象迅速地访问满足某一条件的所有元素.DSM的这种用法反映了关系(限制)作为一

种静态 schema 的传统作用.[7]

2.4 其 它

文献[8]介绍了在 OODB 中一种使用规则来表示和维持系统的一致性的方法,同 DSM 一样,该系统把规则处理同样看作是核心成份,但一个规则引入了一些谓词,例如 DRIVES:(OWN (person,car) (Relate person))定义了一个 DRIVES 谓词,它涉及到了 OWN 谓词、类 person 和 car,谓词可用来产生两个相关对象之间的连结,RELATE 是实现关系 DRIVES 的一个方法. 与 DSM 相比,一个关系不仅用于模式的定义,而且可用于产生实际的连结并动态地检查. 我们认为引入新谓词的做法在 OO 模型中可能会带来概念上和模型上的冗余.

3 限制对象

我们提出利用特定的对象(称为限制对象)来表示对象内部或对象之间的关系,这种关系既可以是结构性的,也可以是控制性的. 限制对象的引入不仅使关系(约束)明显化,而且能动态地增加、删除或修改关系,方便了设计过程. 限制对象的功能包括了检查和问题求解,并由用户控制. 与第 2 节中介绍的几个系统相比,限制对象的概念的突出优点之一在于它无需在 OO 模型下引入另外的程序设计概念,如规则、关系和 Thing,而统一由对象来表示.

3.1 限制对象与一般对象

限制对象是为了让用户有一种手段去描述和控制对象内部及对象之间的结构和行为关系,但我们认为,限制对象的这种手段应该是一般面向对象设计的一种补充,而不是主角(象 Contract 或 Thing 那样).

在一般的面向对象系统中,对象之间的一些结构关系已经通过属性名、子类关系等反映出来,而复杂的行为可以通过复杂的方法来表达,这样设计的软件的效率高,但整个设计需要计划. 为了使复杂问题域中的对象设计更局部化和方便设计过程,一些更复杂的关系可以在已有的但未仔细计划和设计的结构、行为上补充生成,这也是引入限制的原因之一.

对象间的关系又分为结构关系和行为联系. 我们认为前者比后者更需要得到专门机构的控制,因为新的行为联系可以通过修改或重定义新的方法来达到,而结构之间的约束则一般没有直接的方式来控制. 为此,限制对象仅用于描述对象的属性之间的关系限制,而行为(消息)之间的关系由子类重新设计方法体或由 daemon 重解释消息来控制. 这一点是类似于 Thinglab 而不同于 Contract,这种功能也类似于传统框架或语义数据模型中的 daemon 或活动值.[9]

3.2 限制对象的种类

一个限制对象描述了一组对象之间的某种关系,这个限制可以是加在具体的对象上,也可以是适用于很多组的对象,例如适用于一个类的实例之间. 为此我们区分了 3 种不同的限制对象,它们是特例限制对象、抽象限制对象、类限制对象. 我们用 OL—4 语言[10]来描述这些对象.

A. 一个特例限制对象只能作用于一组具体的对象上,这种对象属于类 Scobject (Special—constraint—object),一个典型的这种对象的产生过程是由如下的消息实现:

(send scobject 'new

```
:name 'sco1
:relating—objs '((obj1 iv1) (obj2 iv2 iv3) obj3)
:predicate form)
```

其中 sco1 是这个限制对象的名字,它控制了具体对象 obj1 的属性 iv1、obj2 的属性 iv2 和 iv3、obj3 的所有属性的访问,sco1 的 predicate 属性值 form 给出了这个限制的控制代码,这个代码由 sco1 对象的相应方法激发运行. form 中可出现对象名字 obj1、obj2 和 obj3. sco1 将在对象 obj1 的属性 iv1、obj2 的属性 iv2 和 iv3 或 obj3 的一个属性被访问时起作用.

B. 抽象限制对象是指需要某种形实对象参数结合以后才能起作用的限制对象. 这种对象属于类 Acobject(Abstract constraint object),并可由如下的消息产生:

```
(send acobject 'new
      :name 'aco1
      :obj—variables '(x y)
      :relating—objs '((x iv1) (obj2 iv2 iv3) y)
      :predicate form)
```

这个消息所产生的对象 aco1 和上述 sco1 相比,多了一个属性 obj—variables,该属性专门指明该限制对象中的对象参变量 x 和 y. obj2 是一个具体对象,form 描述了 obj2、x 和 y 三者之间的关系,它们可以出现在 form 中. Acobject 定义了一个方法 instantiate,该方法专门用于实例化一个抽象限制对象,其结果是产生一个特例限制对象. 如消息(send aco1 'instantiate :name 'sco2 :x obj1 :y obj3)产生了另一个作用于 obj1、obj2 和 obj3 的相应属性上的特例控制对象 sco2. 但 sco2 和 sco1 的作用是不完全相同的,sco2 仅当 x 的值 obj1 的属性 iv1 或 y 的值 obj3 的一个属性被访问时才起作用,当 obj2 的属性被访问时不起作用.

C. 类限制对象是指作用于一个类的所有实例上的限制对象,它们属于类 Ccobject (Class—based Constraint Object),产生实例的消息为:

```
(send ccobject 'new
      :name 'cco1
      :relating—class '(cls iv1 iv2)
      :relating—objs '((obj1 iv3) (obj2 iv4 iv5) obj3)
      :predicate form)
```

其中属性 relating—class 给出限制对象 ccob1 所涉及的类 cls 及其实例属性名 iv1 和 iv2. form 中可出现 obj1、obj2 和 obj3,在 FORM 中类 cls 中的一个实例用变量 * obj * 来表示. 类似于 Acobject 的实例,cco1 也仅当类 cls 实例的属性 iv1 或 iv2 被访问时才发生作用,当 obj1、obj2 和 obj3 属性被访问时将不起作用.

上述 3 种限制对象分别又有 3 种子类,它们是读控制对象、写控制对象和结构控制对象. 读控制对象在对象属性被读时起作用;写控制对象和结构控制对象在写操作时发生作用. 它们分别有方法 'read—ok'、'write—ok' 和 'struct—ok',在读写访问时,这些方法分别激发相应的控制对象起作用.

3.3 检查与计算

各种限制对象的 predicate 属性的值 form 是任意的一个程序段,它可以是检查相关对

象之间的关系一致性.如一个人对象的年龄属性必须大于0,但必须小于其父亲的年龄,如果其父亲仍活着.因此,一旦这种关系被一限制对象所规定,则当年龄属性被修改时该限制对象必须检查新的年龄值是否满足上述约束.

限制的另一种功能是在一组相关变量中从已知的变量计算未知的变量值,如下所示:

```
(send scobject 'new
      :name 'sco1
      :relating-objs '((obj1 iv1) (obj2 iv2 iv3))
      :predicate '(and (or (iv-value obj2 iv3)
                           (eq (iv-value obj1 iv1)
                               (iv-value obj2 iv2))
                       (setf (iv-value obj2 iv3) a-new-value)))
```

该对象是指当对象 obj2 的属性 iv3 有值,或 obj1 的 iv1 属性值等于 obj2 的 iv2 属性值时,将 obj2 的 iv3 属性改成新值.

4　实现与问题

以解释系统为例,实现上把各个限制对象和相应的属性名登记在系统的全局属性表中,每次对象访问时,增加对此表的访问判断,如果相应的入口项非空,则对其中的限制对象进行计算后并据计算结果返回对象的属性值.

当限制分散在不同对象中时,带来的一个严重的危险是可能使整个搜索过程变成一个无穷循环,即便当前状态是一个解的情况下也可能发生这种情形,这是因为限制对象的设计有分散性,不同的用户可能对某一对象关系施加多个不同的控制对象;另外由于满足方法的任意性,一个限制对象可以通过新的访问操作激发其它限制对象的作用(甚至再次激发它自己),这个过程中也有可能重新对先前已被"满足"的限制的重检测(或激发),从而可能形成一个无限循环,下面这两个写限制对象将互相激发,从而引起循环:

```
    cobj1
relating-objs: (obj1 iv1)
predicate: ((setf (iv-value obj1 iv2) value1))
    cobj2
relating-objs: (obj1 iv2)
predicate: ((setf (iv-value obj1 iv1) value2))
```

为了防止限制对象的循环激发,系统要有一定的控制手段.一种方法是保存当前求解中已经历过的状态,一旦重复则表示有无限循环,这种方法的依据是:对于各变量均是有限域的求解过程,其无穷循环是可测的,但对于无限域的情形,则没有较简单的检测方法.第二种办法是每次访问时记录已经激发过的限制对象,且规定每个限制对象只能激发一次.这种方法不能保证获得一个解.

5　讨　论

限制对象是一种灵活的控制手段,利用它用户可以在已有的对象系统中方便增加新的

控制代码,而不影响原有程序的总的控制流.显式的限制表示可以增加可理解性.但不当的使用会带来循环问题.本文介绍的限制对象的思想和功能已在面向对象数据库 Daemon4/1 中实现.[10]上面没有涉及的一个内容是限制对象在类结构中的继承,目前在 OL—4 中规定超类上定义的类限制对象是传播到它的子类的.

参考文献

1 Richard Helm *et al*. Contracts: specifying behavioral compositions in object—oriented systems. ACM Sigplan Notices 25:10, Dec. 1990. 169～180.

2 Maloney J *et al*. Constraint technology for user—interface construction in thing Lab II. ACM Sigplan Notices 24: 10 Dec. 1989. 381～387.

3 Freeman—Benson B N. A module mechanism for constraints in smalltalk. ACM Sigplan Notices 24:10 Dec. 1989. 389～396.

4 Borning A *et al*. Constraint hierarchies. ACM Sigplan Notices 22:12, Dec. 1987. 48～60.

5 Shan A V *et al*. DSM: an object—relationship modeling language. ACM Sigplan Notices 24:10, Dec. 1989. 191～202.

6 Kames Rumbaugh. Relations as semantic constructs in an object—oriented language. ACM Sigplan Notices 22:12, Dec. 1987. 466～481.

7 Klas W, Neuhold E J, Schrefl M. Tailoring object—oriented data models through metaclasses. 1st DOOD conference, Kyoto Japan, Dec. 7—9, 1989.

8 Goutas S, Soupos P, Christodoulakis D. A new approach towards an object—oriented data base system. Microprocessing and Microprogramming 27, North—Holland, 1989.

9 Daniel G. Bobrow, Srefik M J. Perspectives on artificial intelligence programming. Science, Feb. 1986, **231**.

10 吴建清.面向对象数据库模型和语言系统的研究[博士论文].国防科技大学计算机系,1992.

USING OBJECTS TO REPRESENT AND COMPUTE CONSTRAINTS

Xi Jianqing Wang Nengbin

(*Department of Computer Science Southeast University Nanjing* 210018)

Abstract Constraints reflect relationships among a group of objects or among p arts of an object. Current object—oriented paradigm does not support the explicitly expressing and computing. This paper introduces a method to explicitly express constraints using objects. The purpose ofit is to keep the language model simple when introducing constraints. Problems are also discussed with possible solutions.

Key words Constraints, constraint satisfaction, object, OOPL, PL.