DOL：一个演绎对象库语言*

王修伦 孙永强

(上海交通大学计算机科学与工程系 上海 200030)

E-mail: wangxiulun@hotmail.com

摘要 演绎对象数据库是对象数据模型和演绎数据库集合的产物.它集成演绎数据库的查询能力和对象数据库的强大建模能力.DOL(deductive object base language)是作者设计的一种演绎对象库语言,它支持类、类层次、继承、集合、部分集、方法及重载和否定.文章着重研究继承、重载和复杂结构化值的交互关系.定义了压缩操作子和重载操作子.基于这两个操作子,定义了与经典逻辑程序类似的直接后承操作子,并研究其不定点性质.

关键词 演绎对象库语言,部分集,压缩操作子,重载操作子,描述型语义.

中图法分类号 TP311

对象数据模型和具有演绎能力的查询语言是传统数据库的两个很重要的扩充.对象数据模型一般包含两类:面向对象数据模型OODM(object-oriented data model)和复杂对象数据模型CODM(complex object data model).OODM吸收面向对象语言的主要概念,每一对象都由不变的对象标识和可变的状态组成.对象的状态由属性和方法来刻画,类作为对象的分类机制,子类关系可以描述继承概念,子类可继承父类的属性和方法;而CODM从关系数据模型发展而来,对象是一种结构化值,可以是嵌套元组和集合,并没有对象标识概念.

将对象数据模型和逻辑语言集成受到许多研究者的重视,出现了一些集成语言,如F-LOGIC[1],IQL[2],ROL[3],Gulog[4]和Reletionlog[5]等.在纯粹的OODM中,为每一对象指派一个对象标识有时并不方便,也是不必要的.而在CODM中,信息共享和语义约束却不如OODM灵活.因而建立能体现CODM和OODM的灵活的数据模型的演绎对象库语言就显得很有意义.DOL(deductive object base language)正是在这方面的一个探索,既支持对象标识、继承、重载,也支持复杂对象.为此,本文着重研究了继承、重载和复杂对象之间的交互关系.

1 语法介绍

为描述DOL的语法,首先给出下面的符号极其相应的意义:(1) 基本型名集合B(例如:*string*,*int*等);(2) 基本类名集合C,与实体关系数据模型类似,C中元素相当于其实体概念,实体中的元素可以用一个符号表示,称为原子对象标识;(3) 关系名集合R,其中元素相当于实体关系数据模型中的关系;(4) 原子对象标识符集合O;(5) 属性名集合A;(6) 方法名集合M;(7) 变量集合V.

定义1.1. 基于B,C和R的类名集Cs定义如下:

(1) $B\subset Cs$,$C\subset Cs$为原子类;

(2) 如果$c\in Cs$,则$\{c\}\in Cs$为集合类;

(3) 如果$r\in R$,$c_1,\ldots,c_n\in Cs$,则$r(c_1,\ldots,c_n)\in Cs$.对Cs的一个限制是,不能有两个同名关系说明.

(4) 如果$a_1,\ldots,a_n\in A$,$c_1,\ldots,c_n\in Cs$,$[a_1:c_1,\ldots,a_n:c_n]\in Cs$为元组类.

定义1.2. 设Ch为Cs的一个子集,表示原子类和关系类的集合,则isa是定义在Ch上的一个偏序关系.对于Ch的任意两个类a,b,a isa b表示a是b的子类,b是a的父类.isa * 为isa的自反、传递闭包.与ROL和LOL[3]类似,可以将isa关系扩展到Cs上,我们用$e_$isa表示,扩展如下:

(1) 如果a isa b,则a $e_$isa b;

---

* 作者王修伦,1970年生,博士,主要研究领域为对象数据库.孙永强,1931年生,教授,博导,主要研究领域为计算机软件理论,新型语言,并行处理.

本文通讯联系人:王修伦,上海 200030,上海交通大学计算机科学与工程系

本文1997-06-09收到原稿,1997-09-19收到修改稿

(2) 如果$\{c\}\in Cs$,$\{c'\}\in Cs$并且c isa c',则$\{c\}$ $e_$isa $\{c'\}$;

(3) 如果$[a_1:c_1,\ldots,a_n:c_n]\in Cs$,$[a_1:C'_1,\ldots,a_n:c'_n]\in Cs$,并且$c'_i$ isa c_i,或$c'_i=c_i$,$i=1,\ldots,n$,则$[a_1:c_1,\ldots,a_n:c_n]$ $e_$isa $[a_1:c'_1,\ldots,a_n:c'_n]$;

显然,$e_$isa关系也具有自反性、反对称性和传递性.

定义 1.3. 属性和方法. $c[a\Rightarrow c_1]$为类的属性说明,a为c的一个属性名.$c[m@c_1,\ldots,c_n\rightarrow c_{n+1}]$为类的方法基调说明,$m$为$c$的方法名,$c_1,\ldots,c_n,c_{n+1}\in Cs$,前$n$个为$m$的参数类型,$c_{n+1}$为结果类型,若$n=0$,可简写为$[m\Rightarrow c_1]$.

基于上面的定义,下面给出数据库模式概念.

定义 1.4. DOL数据库模式为一个五元组$K=(Cs,A,M,\text{isa},Dc)$,其中Cs,A,M,isa的意义如上定义,isa说明的是类间直接子类关系,Dc为DOL中属性和方法说明集合.

DOL允许多继承,然而多继承也将导致继承的二义性.关于继承的二义性,已有许多文献对此作了分析,并提出多种解决办法.在此,我们不讨论这个问题,只对模式说明作一些限制,以保证多继承无二义性.

定义 1.5. $K=(Cs,A,M,\text{isa},Dc)$为一种数据库模式,我们定义

$$Dc*=\{c[m@c_1,\ldots,c_n\Rightarrow c_{n+1}]\mid \exists\, c\in Cs, c \text{ isa}* c_1, c_1[m@c_1,\ldots,c_n\Rightarrow c_{n+1}]\in Dc.$$

数据库模式K是无二义性的,如果对于$Dc*$中任意的、形如$c[m@c_1,\ldots,c_n\Rightarrow c_{n+1}]$,$c[m@c'_1,\ldots,c'_n\Rightarrow c'_{n+1}]$的模式说明,则在$Dc*$中存在形如$c[m@c''_1,\ldots,c''_n\Rightarrow c''_{n-1}]$的模式说明,满足$c'';e_\text{isa}* c_i;c''_i e_\text{isa}* c'_i$.

在下面的讨论中,我们假设数据库模式是无二义性的.

定义 1.6. K为数据库模式,O为对象标识集,D为B所对应的值域,则在DOL中,"项"可定义如下:

(1) D,C中的元素分别为值项和对象标识项;

(2) $r\in R,o_1,\ldots,o_n$为项,则$r(o_1,\ldots,o_n)$为关系项;

(3) 如果$o_1,\ldots,o_n$为项,$a_1,\ldots,a_n\in A$为属性名,且互不相同,则$[a_1\rightarrow o_1,\ldots,a_n\rightarrow o_n]$为元组项;

(4) 如果$o_1,\ldots,o_n$为项,则$\langle o_1,\ldots,o_n\rangle$表示部分集合项,部分集表示一个集合的部分信息;

(5) 如果$o_1,\ldots,o_n$为项,则$\{o_1,\ldots,o_n\}$为集合项.如果$o_1,\ldots,o_n$为不包含部分集,则$\{o_1,\ldots,o_n\}$为完全集合项;

(6) 变量是表示值项.对象标识项和集合项的变元项.

部分集首先由Liu在Relationlog[5]中提出,它用来有效地处理集合关系;本文引入部分集,是为了更加灵活地支持对象方法表达式、增强系统建模能力.不含变量的项为基项,基项表示一个对象.由于引入元组、关系和集合构造子,使得对象具有复杂结构,复杂对象将影响到程序的语义.

定义 1.7. DOL中的对象表达式可以递归定义如下:

(1) o为对象标识符,c为类名,则$o:c$为简单对象表达式(SOE),在DOL中,每一对象标识o,只能存在一个类c,满足o为c的直接实例,我们用Class(o)表示;

(2) 如果o为关系项,则o为简单对象表达式;

(3) o_1,o_2为项,a为属性,则$o_1[a\rightarrow o_2]$为简单对象表达式;

(4) $o:c$为简单对象表达式,$o_1,\ldots,o_n,o_{n+1}$为项,则$o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$为对象函数方法表达式.$c::o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$为类型化方法表达式;

(5) $o:c,o[a_1\rightarrow o_1],\ldots,o[a_n\rightarrow o_n]$为对象表达式,则$o[a_1\rightarrow o_1,\ldots,a_n\rightarrow o_n]$为复合对象表达式.

为了使问题变得简洁,我们限定部分集不能出现在方法对象表达式的参数中.

定义 1.8. DOL规则.在DOL中有两类规则,一种与通常演绎数据库中规则相同,另一种为DOL所特有,称为对象方法定义规则.它们的形式都为$H\leftarrow L_1,\ldots,L_n$,其中$H,L_1,\ldots,L_n$都是对象表达式.但是,我们对这两种规则作一个限制:即如果一个规则为方法定义规则时,规则头的方法表达式必须是带类型的,H的形式为$c::X[m@o_1,\ldots,o_n\rightarrow o_n+1]$,表示方法$m$在类$c$上的定义.这与Datalog$^{\text{meth}}$[6]和Gulog[4]类似.

2 DOL语义

本节进一步研究DOL的描述型语义,我们的语义模型与经典逻辑程序类似,也是基于Herbrand模型理论.下面首先给出Herbrand域和基的概念,然后讨论模型的定义和性质,重点分析部分集和方法规则的重载.

定义 2.1. 我们定义Herbrand域U_K的概念,在语法部分,我们给出项的概念,对于一个数据库K,U_K为其所有项集.

上述定义保证集合只能是有限元素集,而不能是无限元素,但集合可以是嵌套的,这与LDL[7]对集合的支持是不

同的,有限性使得DOL的语义具有简洁性.

定义 2.2. Herbrand 基 B_K 为所有基简单对象表达式集合.在 B_K 中,我们并没有包含组合对象表达式,这是由于任一组合对象表达式所表达的信息等价于其组成对象表达式所表达的信息之和.

定义 2.3. B_K 的一个子集称为是压缩的(Compact),如果它包含的对象表达式中没有部分集出现.基于模式 K 的一个解释 I 为 B_K 的一个压缩子集,一个解释称为是带类型的,如果方法表达式都是带类型的.

为了讨论规则和程序在解释 I 下满足,我们先定义下面的一些概念:

定义 2.4. S_B 为 B_K 的一个子集,类在 S_B 中的实例定义如下:

(1) 一个值为 B 中值类的实例,如果这个值在该值类所表示的域中;

(2) 一个对象标识符 o 为类 c 的实例,如果 $o:c\in S_B$;

(3) 部分集或完全集 s 为 $\{c\}$ 的实例,当且仅当 $\forall\, o\in s$,o 为 c 的实例;

(4) 关系对象 $r(o_1,\ldots,o_n)$ 为类 $r(c_1,\ldots,c_n)$ 的实例,当且仅当 o_i 为 c_i 的实例,$i=1,\ldots,n$;

(5) 元组对象 $[a_1\rightarrow o_1,\ldots,a_n\rightarrow o_n]$ 为 $[a_1\rightarrow c_1,\ldots,a_n\rightarrow c_n]$ 类的实例,当且仅当 o_i 为 c_i 的实例,$i=1,\ldots,n$.

定义 2.5. K 为一个数据库模式.S_B 为 B_K 的一个子集,φ 为一个基简单对象表达式,则 φ 在 S_B 中是良类型的,如果下列条件之一成立.

(1) $\varphi=o:c$,且 o 为 c 在 S_B 中的实例;

(2) $\varphi=o[a\rightarrow o_1]$,则$\exists\ c[a\Rightarrow c_1]\in K$,使得 o,o_1 分别为 c,c_1 在 S_B 中的实例;

(3) $\varphi=o:c[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$,则 $c[m@c_1,\ldots,c_n\Rightarrow c_{n-1}]\in K$,并且 $o,o_1,\ldots o_n,o_{n+1}$ 分别为 $c,c_1,\ldots,c_n,c_{n+1}$ 在 S_B 中的实例;

(4) $\varphi=r(o_1,\ldots,o_n)$,则$\exists\ r(c_1,\ldots,c_n)\in K$,当 o_i 为 c_i 的实例,$i=1,\ldots,n$.

定义 2.6. 置换 θ 为变量集 V 到 U_K 的映射,它的意义与经典逻辑中的置换一样.可将置换应用到项、对象表达式和规则上.

前面我们引入的部分集概念表示集合信息的一个部分信息,推而广之,每一对象都可由其部分对象表示,表示对象的不完全信息,因而有如下定义.

定义 2.7. 一个对象 o 为压缩对象 o' 的部分对象,表示为 $o\leqslant o'$,当且仅当

(1) $o=o'$;

(2) $o=r(o_1,\ldots,o_n)$,$o'=r(o'_1,\ldots,o'_n)$,并且 $o_i\leqslant o'_i(i=1,\ldots,n)$;

(3) $o=[a_1\rightarrow o_1,\ldots,a_n\rightarrow o_n]$,$o'=[a_1\rightarrow o'_1,\ldots,a_n\rightarrow o'_n]$,并且 $o_i\leqslant o'_i(i=1,\ldots,n)$;

(4) o 为部分集,o' 为完全集,且$\forall\ o_i\in o$,$\exists\ o'_i\in o'$满足 $o_i\leqslant o'$.

同样,可将这个概念扩展到对象表达式上,下面,我们只给出对象函数方法表达式的定义,其余同.$o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]\leqslant o[m@o_1,\ldots,o_n\rightarrow o'_{n+1}]$,如果 $o_{n+1}\leqslant o'_{n+1}$.

在对象语言中,支持方法的好处是提供对象的动态性质,另一方面也为缺省表示带来方便,然而这也导致方法重载的存在.1993年,Abiteboul[6]讨论了继承与重载的关系,对重载的两种形式:静态与动态重载进行分析,并给出相应的变换方法.静态重载是指方法对该类上的对象是全映射,即该类对象对方法的调用总是能够成功的;动态重载是指方法对该类上的对象是部分映射,即该类对象对方法的调用并不总是能够成功的,当失败时,继承起相应父类中的方法,因而起到动态缺省机制的作用.

随着继承的动态性将产生一个问题:当一个类有多个父类时,子类将可能有多个父类中的方法被动态继承,因此,需提供一种选择机制.我们用 $\mathrm{Lim}(c,c')$ 表示 c' 为 c 的重载父类;下面,我们从语义角度来分析方法重载.如果有两个方法定义规则:

$r=c::X[m@o_1,\ldots,o_n\rightarrow o_{n+1}]\leftarrow B$,

$r'=c'::X[m@o_1,\ldots,o_n\rightarrow o'_{n+1}]\leftarrow B'$,

如果存在两个基置换 θ,θ',使得 C 与 C' 的两个基实例子句满足下面两个条件之一:

(1) 如果 $\mathrm{Lim}(c,c')$,则 $\theta(X)=\theta'(X)$,$\theta'(o'_i)=\theta(o_i)\quad(i=1,\ldots,n)$,$\theta'(o'_{n+1})\neq\theta(o_{n+1})$,

(2) 否则,$\theta(X)=\theta'(X)$,$\theta'(o'_i)=\theta(o_i)(i=1,\ldots,n)$,$\theta'(o'_{n+1})\neq\theta(o_{n+1})$,$\mathrm{Class}(\theta(X))$ isa * c,$\mathrm{Class}(\theta(X))$ isa * c',并且满足 $\mathrm{Lim}(\mathrm{Class}(\theta(X)),c)$,

那么 $C\theta$ 重载 $C'\theta'$.

定义 2.8.[3] 对象间的序关系 $\leqslant_p$.对于任意两个对象 o,o',$o\leqslant_p o'$ 成立,

(1) 如果 $o=o'$;

(2) $o=r(o_1,\ldots,o_n)$, $o'=r(o'_1,\ldots,o'_n)$,并且 $o_i\leqslant_p o'_i(i=1,\ldots,n)$;

(3) $o=[a_1\rightarrow o_1,\ldots,a_n\rightarrow o_n]$, $o'=[a_1\rightarrow o'_1,\ldots,a_n\rightarrow o'_n]$,并且 $o_i\leqslant_p o'_i(i=1,\ldots,n)$;

(4) o,o'为集合项(或为部分集), $o\leqslant_p o'$当且仅当$\forall\ o_i\in o-o'$, $\exists\ o'_i\in o'-o$,满足 $o_i\leqslant_p o'_i$.

可以将这种序关系进一步扩展到压缩对象表达式上来.

(1) $o_1;c_1\leqslant_p o'_2;c_2$,当且仅当 $o_1\leqslant_p o'_2$;

(2) $o[a\rightarrow o_1]\leqslant_p o'[a\rightarrow o'_1]$当且仅当 $o\leqslant_p o'$, $o_1\leqslant_p o'_1$;

(3) $r(o_1,\ldots,o_n)\leqslant_p r(o'_1,\ldots,o'_n)$,当且仅当 $o_i\leqslant_p o'_i(i=1,\ldots,n)$;

(4) $o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]\leqslant_p o[m@o_1,\ldots,o_n\rightarrow o'_{n+1}]$,当且仅当 $o_{n+1}\leqslant_p o'_{n+1}$.

命题 2.1.[3] $\leqslant_p$是偏序关系.

在上述的序关系和所定义的一些概念的基础上,给出解释满足($\vdash$)的定义:

定义 2.9. K 为一个数据库模式, I 为良类型解释,满足($\vdash$)可定义如下:

(1) o 为 c 的对象,则 $I\vdash o:c$,当且仅当 o 为 I 中 c 的实例,且$\forall\ c'$, c isa* c', $I\vdash o:c'$;

(2) 对于一个形如 $o[a\rightarrow o_1]$(或 $o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$)的对象表达式, $I\vdash o[a\rightarrow o_1]$(或 $o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$),当且仅当存在一个对象表达式 $\varphi\in I$,并且 $o[a\rightarrow o_1]\leqslant_p\varphi$(或 $o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]\leqslant_p\varphi$);

(3) $I\vdash r(o_1,\ldots,o_n)$,当且仅当存在一个对象表达式 $\varphi\in I$,并且 $r(o_1,\ldots,o_n)\leqslant_p\varphi$;

(4) 对于每一复合对象表达式 α, $I\vdash\alpha$,当且仅当 α 的每一组成对象表达式 φ, $I\vdash\varphi$;

(5) 对于比较表达式和集合运算,与经典逻辑相同;

(6) $I\vdash\neg\alpha$,当且仅当 $I\vdash\alpha$ 不成立;

(7) 对于任一规则的实例 r, $I\vdash r$,当且仅当或者对于 r 中的每一对象表达式 I 都满足,或者存在另一实例规则 r', $I\vdash r'$, r'重载 r;

(8) 定义一个程序 P,如果其所有实例规则都满足(7),则 $I\vdash P$.

对于一个数据库模式 K 及程序 P,若 $I\vdash P$,则 I 为 P 的一个模型.

与经典逻辑程序语义一样,DOL 语言的模型并不具有唯一性.为了描述我们所特指的模型,我们需要定义模型间的序关系.

定义 2.10. I_1,I_2 为两个解释, I_1 优选于 I_2(表示为 $I_1\subseteq_p I_2$),如果$\forall\ a\in I_1-I_2$, $\exists\ b\in I_2-I_1$,使得 $a\leqslant_p b$ 成立.

命题 2.2.[3] $\subseteq_p$是偏序关系.

定义 2.11. 一个程序 P 的模型 M 为极小的,当且仅当对于任意 P 的模型 N,如果 $N\subseteq_p M$,则 $N=M$.

3 不动点语义

并不是所有程序都存在模型.本节对程序作一个适当的限制:可层次化,通过引入压缩操作子 *Comp* 和重载操作子 *Ord* 定义程序的直接后承操作子,使得程序极小模型可由一系列这样的操作子计算而得.

我们首先讨论压缩操作子 *Comp*,需要引入下面几个概念.

定义 3.1. DOL 程序的可层次化概念与 ROL 相同,随后我们假定程序是可层次化的.

定义 3.2. 一个集合 S(对象)是同构的,如果 S 为类 $\{c\}$ 的实例. S 是可比较的,如果$\forall\ o,o'\in S$,下列几种情况有一个成立.

(1) $o=o'$.

(2) o,o'是部分集,$\forall\ o_i\in o$, $o'_i\in o'$, o_i,o'_i 是可比较的.

(3) o 为部分集, o' 为完全集,且 $o\leqslant o'$.

可以将可比较概念推广到对象表达式上来,这里仅对对象方法表达式进行说明.对于任意两个形式为 $o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$, $o[m@o_1,\ldots,o_n\rightarrow o'_{n+1}]$的方法表达式,它们是可比较的,iff o_{n+1},o'_{n+1}是可比较的.

压缩操作子 *Comp* 在对象集合 S 上递归,对于如下:

(1) $Comp(\{o\})=o$,如果 o 为压缩的;

(2) S 为一可比较集, o 为 S 中一压缩对象,如果 S 中任意一个元素都是部分对象,则 $Comp(S)=\{o\}\cup Comp(S-\{o\})$.否则, $Comp(S)=und$(表无定义);

(3) S 为部分集, $S''=\{o\mid o\in S',S'\in S\}$,如果 $Comp(S'')$有定义,则 $Comp(S)=\{Comp(S'')\}$,否则, $Comp(S)=und$.

按同样的方法可将 *Comp* 扩展到对象表达式上来.

(1) S 为形如 $o:c$ 的对象表达式集,$S'=\{o\mid o:c\in S\}$,则 $Comp(S)=\{Comp(S'):c\}$;

(2) S 为形如 $o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$的对象方法表达式集,且 $o,o_1,\ldots,o_n$ 为压缩形式,则

$$Comp(S)=o[m@o_1,\ldots,o_n\rightarrow Comp(S')],\text{其中 }S'=\{o_{n+1}\mid o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]\in S\};$$

(3) S 为 $o[a\rightarrow o_1]$形式对象表达式集合,$Comp(S)$的定义与(2)类似;

(4) S 为 $r(o_1,\ldots,o_n)$形式对象表达式集合,$S_i=\{o_i\mid r(o_1,\ldots,o_n)\in S\}$,$Comp(S)=r(Comp(o_1),\ldots,Comp(o_n))$;

(5) S 为任意基对象表达式集合,则 S 可划分为不相交的可比较子集 S_i,$Comp(S)=\bigcup Comp(S_i)$.

命题 3.1. P 为数据库模式 K 的一个程序,M 为 P 的一个模型,则 $Comp(M)=M$.

下面我们引入另一个操作子 *Ord*,来刻画继承和重载问题.在前面我们已经给出两个基对象方法定义规则的重载定义,我们在此基础上定义重载操作子 *Ord*.

我们需定义一个带类型方法表达式,$o[m@o_1:\ldots,o_n\rightarrow o_{n+1}]$为一个方法表达式,则 $c::o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]$为带类型的方法表达式,表示对象 o 调用在类 c 中的方法 m 的定义规则产生的结果.

定义 3.3. 设 S 为一个带类型的对象方法表达式集合,指的是其中的方法表达式可能是带类型的.Untype(S)表示 S 所对应的无类型解释.*Ord* 是一个带类型解释到无类型解释的映射,可以定义为:

$$Ord(S)=\text{Untype}(S')$$

$$S'=S-\{c::o[m@o_1,\ldots,o_n\rightarrow o'_{n+1}]\mid\forall\ c'::o[m@o_1,\ldots,o_n\rightarrow o_{n+1}]\in S,$$
$$\exists\ c::o[m@o_1,\ldots,o_n\rightarrow o'_{n+1}]\in S,\text{使得 }o_{n+1}\neq o'_{n+1},\text{Lim}(c',c)\text{或者}$$
$$\exists\ c''::o[m@o_1,\ldots,o_n\rightarrow o'_{n+1}]\in S,\text{使得 }o_{n-1}\neq o'_{n+1},\text{Lim}(\text{Class}(o),c'')\}$$

Ord 可以扩展到任意基对象表达式集合上来,其扩展的定义是简单的,我们并不加以讨论.

下面定义的模型间的映射函数是基于解释的(可能包含类型信息),随后可看到带类型解释并不会复杂化 Bottom up 语义.设 I 为一个解释,映射函数 OT_P 定义如下:

$OT_P'(I)=\{B\mid\forall\ H:-\text{Body}\in P,\theta$ 为一个置换,B 为 $H\theta$ 的组成对象表达式,如果 B 为方法表达式,则为带类型的,使得 $\text{Untype}(I)\vdash\text{Body}\}$,

$OT_P''(I)=\{o:c\mid o:c'\mid c'\ \text{isa}*\ c\}$,

$OT_P(I)=\text{Comp}(\text{Ord}(OT(I'_P))\cup OT_P''(I)$.

定理 3.1. P 为一个模式 K 的程序,如果 M 为其一个模型,则 $OT_P(M)\subseteq_P M$.

证明:设 $\varphi\in OT_P(M)$,我们分两种情况讨论:如果 φ 包含集合项,则$\exists\ \varphi'_1,\ldots,\varphi'_n\in OT_P'(M),\varphi'_i(i=1,\ldots,n)$为 φ 的子对象,P 中存在属于某个类的 n 个实例规则 $H_i:-\text{Body}_i$,满足 $M\vdash\text{Body}_i,(i=1,\ldots,n),\varphi=H_i$.由于操作子 *Ord* 的作用,在 P 中不会存在重载这个规则的其它实例规则,M 为 P 的模型,由满足的定义,必有 $M\vdash\varphi$ 成立.否则,φ 为对象描述或对象单值属性说明,显然两个操作子并不影响它们的形式,有 $M\vdash\varphi$. □

定义 3.4. 若 $OT_P(I)=I$,则 I 为 OT_P 的一个定点.如果不存在其它定点 I',使得 $I'\subseteq_P I$ 成立,则 I 为 OT_P 的极小定点.

定义 3.5. 设 $OT_1,\ldots,OT_n$ 为操作子序列.其迭代幂定义如下:

$$M_0=\varphi_I$$
$$M_1=OT_1\uparrow^{\omega}(M_0);$$
$$M_2=OT_2\uparrow^{\omega}(M_1);$$
$$\cdots\cdots$$
$$M_n=OT_n\uparrow^{\omega}M_{n-1})=M_P.$$

定理 3.2. 设 $P=p_1,\ldots,p_n$ 为一个可层次化程序,$OT_1,\ldots,OT_n$ 为相应的后承操作子,如果 M_P 有定义,则它是程序的极小模型.

限于篇幅,证明从略.

4 结 论

本文给出了演绎对象数据库语言 DOL 的语法和语义,它体现了当前几种演绎型对象数据库语言的主要特征,如 IQL,F-LOGIC,Gulog 和 ROL 等.F-LOGIC 是对象与框架模型集成的逻辑系统,具有非常灵活的建模能力,然而,其不足之处除一般文献所讨论到的以外,它并不能很好地支持集合语义、复杂对象,在重载方面也只是部分支持.ROL

中对部分集进行了详细的研究,然而它对对象中重要特征继承只是部分支持,而对于方法和重载则未加以考虑. Datalog^{METH}和 Gulog 在 Datalog 的基础上讨论了方法和重载问题,动态重载也是由前者首次提出,但它们的讨论都是只考虑单继承模式,对对象的其他特征是不支持的,它们的建模能力也是很有限的. DOL 语言在复杂对象模型和对象特征上给予较全面的支持,复杂对象、集合处理和 OO 中的方法,继承及重载都获得一致的处理,是一个功能很强大的演绎对象库语言.

参考文献

1 Kifer M, Lausen G, Wu J. Logic foundations of object oriented and frame based language. Journal of ACM, 1995,42(4):741~843

2 Abiteboul S, Kanellakis P C. Object identify as a query language primitive. In: Proceedings of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1989. 157~179

3 Liu Meng-chi. ROL: the deductive object base programming language. Information Systems, 1996,21(5):431~457

4 Dobbie G, Topor. On the declarative and procedural semantics of deductive object oriented systems. Journal of Intelligent Information Systems, 1995,4(2):193~219

5 Liu Meng-chi. Relationlog: a typed extension to datalog with sets and tuples. In: Proceedings of the International Logic Programming Symposium. MIT Press, December 1995. 83~97

6 Abiteboul S, Lausen G, Uphoff H *et al*. Methods and rules. In: Proceedings of ACM SIGMOD International Conference on Management of Data. New York, 1993. 32~41

7 Beeri C, Naqvi S, Shmuel O *et al*. Set construction in a logic database language. Journal of Logic Programming, 1991,10(3,4):181~232

DOL: A Deductive Object Base Language

WANG Xiu-lun SUN Yong-qiang

(*Department of Computer Science and Engineering Shanghai Jiaotong University Shanghai* 200030)

Abstract The integration of deductive database and object database results in deductive object database which has capacities of deductive query of deductive database and strong modeling of object database. DOL (deductive object base language) is designed as a deductive object base language. It supports class, object identifier, complex object, partial set, inheritance and overriding. A immediate consequent operator, similar to tradition's, is defined based on two operators: compact operator and overriding operator in this paper. In addition, a fixed point semantics is analyzed.

Key words Deductive object database, partial set, compact operator, overriding operator, declarative semantics.