对象行为等价的终结共代数语义

余珊珊[1] 李师贤[1] 苏锦钿[2]
(中山大学信息科学与技术学院 广州 510275)[1] (华南理工大学计算机科学与工程学院 广州 510640)[2]

摘 要 终结共代数上的互模拟是等价关系,这一性质为对象的行为等价提供了一种基于共归纳原理的证明方法。首先,利用共代数给出面向对象方法中的抽象类、类和对象的形式化描述,其中抽象类被定义为一个包含方法和断言声明的类规范,类被定义为满足类规范的共代数,类的各个对象看成是共代数状态空间上的元素,而对象中方法的各种行为结构则通过强 Monads 进行参数化描述;接着,利用类规范的终结共代数给出对象行为等价关系的证明方法以及在各种不同 Monads 结构下的终结共代数语义;最后,通过实例说明如何利用 PVS 工具对研究结果进行验证。

关键词 对象,行为等价,共代数方法,终结共代数,强 Monads
中图法分类号 TP301.2 **文献标识码** A

Final Coalgebraic Semantics for Behavioral Equality of Objects

YU Shan-shan[1] LI Shi-xian[1] SU Jin-dian[2]
(School of Information Science and Technology, Sun Yet-Sen University, Guangzhou 510275, China)[1]
(College of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China)[2]

Abstract The equality characteristics of bisimulations on final coalgebras entail a proof method for behavioral equality of objects based on coinductive principles. Firstly, we used coalgebras to give formal descriptions about abstract classes, classes and objects in object oriented methods, where abstract class was defined as a class specification including declarations of methods and assertions, and classes satisfying the class specification were described as coalgebras. Each object belonging to a class was viewed as an element of the state space of the class, as coalgebras. Various behavioral structures of methods in objects were described parametrically by strong Monads. Secondly, we used final coalgebras of class specification to give a proof method for objects' behavioral equality and their final coalgebraic semantics under the conditions of different Monads. Finally, some examples were employed to demonstrate how to use PVS tools for verifications.

Keywords Objects, Behavioral equivalence, Coalgebraic methods, Final coalgebras, Strong Monads

1 引言

在面向对象方法的研究中,对象常被看成是一个封装了内部状态的黑盒子,用户只能通过对象的外部可观察行为来了解其内部状态的变化。因此,行为等价自然地成为了比较不同对象间行为关系的一种主要方法,并引起了许多学者的关注与研究。例如,M. Papathomas 在其博士论文[1]中根据 CCS(Calculus of Communicating Systems)中进程的等价性分析了对象和类之间的行为等价及可替换性。N. Hameurlain 等利用 Petri 网研究了对象间的行为类型和行为子类型[2]。C. Peter 等人利用并发对象演算(Concurrent Object Calculus)对并发系统中对象间的顺序消息传递进行分析,并定义了对象上的行为同余关系和结构等价关系[3]。A. Goel 通过标号转换系统给出交互式系统中的进程类(Process Class)和主动对象等的形式化描述,并研究其行为等价关系[4]。R. Burstall 等则从代数的角度利用带隐藏类别(Hidden Sorts)的基调描述面向对象方法,并定义了所谓的行为代数(Behavioral Algebras)及其终结行为代数来研究对象的行为等价关系[5]。随后 M. Berrima 等进一步对代数观察等价与互模拟之间的关系进行研究,并探讨了对象的观察等价关系[6]。

由于进程代数、Petri 网或标号转换系统等形式化方法倾向于将对象看成是进程,并利用互模拟理论从纯行为的角度(即忽略其内部状态)考察对象的行为等价关系,因此它们在刻画对象的内部状态与外部行为之间的关系方面存在一定的不足。而代数方法主要是从语法构造的角度描述系统的归纳构造,不适合于描述像对象这一类基于状态的系统的(无限)动态行为。

作为代数的范畴对偶概念,共代数(Coalgebras)是近年来计算机科学基础理论中的一个新兴研究方向,已经成功地应用于状态转换系统、自动机、进程代数、并发程序语义和面向对象形式语义等领域,并被证明为可以构成状态转换系统、进程代数和自动机等形式化方法的理论基础[7,8]。

到稿日期:2011-03-06 返修日期:2011-06-03 本文受国家自然科学基金项目(60673122),广东省自然科学基金项目(8151030007000002)资助。

余珊珊(1980—),女,博士生,CCF 会员,主要研究领域为形式语义、软件工程等,E-mail:yushsh@mail.sysu.edu.cn;李师贤(1944—),男,教授,博士生导师,主要研究领域为软件工程、分布计算。

对象本身就是一种基于状态的系统，因此非常适合利用共代数作为其数学理论基础。H. Reichel 最早在文献[9]中指出终结共代数(Final Coalgebras)可以作为面向对象语言的形式化语义基础。随后，B. Jacobs 及其学生在这一领域做出许多贡献。例如，他们利用共代数给出抽象类、类和对象的形式化描述[10]，进而研究继承和精化关系等重要的面向对象特征[11,12]。同时，B. Jacobs 等人还利用共代数互模拟及终结共代数对对象的行为等价关系进行深入的研究[10,12]。但这些研究主要针对特定函子下的对象行为关系，没有给出对象中方法的各种不同行为模式的统一描述及在终结语义下的数学解释。

因此，本文的主要工作是在上述研究的基础上利用强 Monads 给出对象中方法的各种行为结构的统一描述，并进一步研究对象行为等价的终结共代数语义。本文第 2 节利用共代数给出抽象类、类和对象的形式化描述；第 3 节分析不同强 Monads 结构下的对象行为等价关系的终结共代数语义；第 4 节通过实例说明如何利用 PVS 工具进行验证；最后进行总结并给出下一步的工作。

2 抽象类、类和对象的共代数描述

本文假设读者具有一些共代数和范畴论的知识，关于共代数的详细介绍可参考文献[7,13]。

在共代数的视角下，抽象类可视为一个类规范。

定义 1 一个类规范(Class Specification)定义为一个四元组 $\Gamma=\{X,F,A,C\}$，其中：

(1)X 是由类规范中所有的公有及私有变量的类型所构成的状态空间，假定相同名称的变量具有相同的类型；

(2)F 为集合范畴上的自函子 $F:Set\to Set$，是由类规范中所有的属性方法和操作方法所构成的有限一元多项式函子；

(3)A 是一个有限公理(或称为断言)集合，用于描述 F 中方法的行为约束关系，即给出方法在执行过程中必须满足的行为语义；

(4)C 是一个有限公理集合，描述了类在初始化时需要满足的条件。

函子 F 中包括两种不同类型的方法：属性方法 $at:X\times I'\to O'$ 和操作方法 $op:X\times I\to X\times O$。$at$ 给出了对 X 的一个观察值，其执行不会改变 X，即 $\forall i'\in I'.\ \forall x\in X.\ x.\ at(i')=at(x)(i')\in O$。$op$ 的执行将改变 X 的内部状态，并可能输出结果，即有 $\forall i\in I.\ x.\ op(i)=op(x,i)\in X\times O$。若 F 包含多个属性方法，可将 at 看成是 $X\to\prod_{1\leqslant i\leqslant n}O_i'^{I_i'}$ 或 $\prod_{1\leqslant i\leqslant n}(X\times I_i')\to\prod_{1\leqslant i\leqslant n}O_i'$ 的形式。若 at 和 op 包含多个参数 $X\times I_1\times\cdots\times I_n$ 或 $X\times O_1\times\cdots\times O_n$，可将 I'，I，O 和 O' 看成是积 $I_1\times\cdots\times I_n$ 和 $O_1\times\cdots\times O_n$ 的形式。若 F 包含多个操作方法，可将 op 表示为 $X\to\prod_{1\leqslant j\leqslant m}(X\times O_j)^{I_j}$ 或 $\prod_{1\leqslant j\leqslant m}(X\times I_j)\to\prod_{1\leqslant j\leqslant m}O_j$。利用 Currying 操作和积操作可将 at 和 op 合并为：$\alpha=\langle at,op\rangle:X\to O'^{I'}\times(X\times O)^I$。

一个类规范实际上相当于一个抽象类或接口，给出了方法的声明及其行为语义约束，而不涉及到具体的实现。

给定类规范 $\Gamma=\{X,F,A,C\}$，若 F 在状态 $x(x\in X)$ 下满足 Γ 中的公理集合 A，则记为 $F,x|=A$；若对所有的 $x\in X$，都满足 $F,x|=A$，则称 F 满足 A，记为 $F|=A$。若 F 具有初始状态 $x_0\in X$，且满足 $F|=A$ 和 $F,x_0|=A$，则称 (X,α,x_0) 是满足 Γ 的一个共代数模型，记为 $(X,\alpha,x_0)|=\Gamma$。若 Γ 至少存在一个共代数模型 $(X,\alpha,x_0)|=\Gamma$，则称 Γ 为一致的(Consistent)。

根据可见性(Visibility)的不同，可将函子 F 进一步分为 $F=F_{Pub}\times F_{Pri}$，其中 F_{Pub} 和 F_{Pri} 分别表示可见性为 Public 和 Private 的属性方法及操作方法的集合(为了简单起见，本文不讨论 Protected 方法)。

由类规范 $\Gamma=\{X,F,A,C\}$ 可以确定一个抽象类或接口 $(X,\alpha:X\to F(X))$，其中基调给出了函子 F 的具体实现，并且 $(X,\alpha)|=A$。若 (X,α) 具有初始状态 x_0，则 (X,α,x_0) 同时也必须满足 C 中的公理。

为了统一描述 op 的各种不同行为结构(例如确定性、异常、部分或不确定性等)，可利用函数式程序语言(如 Haskell)中的强 Monads[14] 将 $F(X)=O'^{I'}\times(X\times O)^I$ 表示为 $F(X)=O'^{I'}\times B(X\times O)^I$。这里的 B 是从各种具体的行为中抽象出来的强 Monads 结构 (B,η,μ,lt,rt)，其中 (B,η,μ) 表示一个 Monad 结构，强自然转换关系 $lt_{X,Y}:X\times B(Y)\Rightarrow B(X\times Y)$ 和 $rt_{X,Y}:B(X)\times X\Rightarrow B(X\times Y)$ 分别称为左和右强度，用于提供环境相关的信息。对 B 进行实例化就可得到各种不同的行为结构(见表 1)。

表 1 强 Monads B 与行为间的对应关系

Monads 结构	行为模式
B=Id	表示确定性行为
B=Id+1	表示部分行为，可能存在死锁
B=Id+E	表示结果可能为异常行为 E
B=**P**	表示非确定性行为
B=$\mathbf{P}_\omega$	表示有限非确定性行为
B=Id*	表示有序非确定性行为

针对类规范所提供的具体实现就构成了一个类，即相当于抽象类的一个子类。

定义 2 类是满足某个类规范 $\Gamma=\{X,F,A,C\}$ 的一个具体实现，表示为一个共代数模型 $C=(U,\alpha:U\to F(U),u_0\in U)$，其中：

(1)载体集 U 为有限的状态空间，给出了 Γ 中 X 的具体解释；

(2)基调 α 给出了类规范中函子 F 在 U 上的具体实现，并且满足 Γ 中的公理集合 A；

(3)初始状态 $u_0\in U$ 给出了类的构造行为的解释，且 u_0 满足 Γ 中的初始化条件 C，记为 $C(u_0)$。

满足类规范 Γ 的类 C 描述了 Γ 的一种实现语义信息，可简单记为 $C=(U,\alpha,u_0)|=\Gamma$ 或 $(U,\alpha)|=\Gamma$。从面向对象的角度看，C 是由 Γ 所确定的抽象类的一个子类。

定义 3 给定一个类 $C=(U,\alpha,u_0)|=\Gamma$，C 的对象 $o\in C$ 定义为 $o=(u\in U,\alpha)$，其中 u 为 C 的状态空间 U 上的某一个元素，称为对象 o 的状态。

例 1 对于一个银行帐号的抽象类，用共代数类规范语言 CCSL(Coalgebraic Class Specification Language)[15] 描述如下：

```
Begin Account:ClassSpec
  Public Method
    balance:Self -> nat; %帐户余额
    deposit:[Self,nat] -> Self;%存款
    debit:[Self,nat]-> Self;%借款
  Assertion
```

```
    SelfVar s:Self
    dep_bal:Forall(i:nat):s. deposit(i). balance= s. balance+i;
    deb_bal:Forall(i:nat):if s. balance >=i then s. debit(i). balance
          =s. balance-i
          else s. debit(i). balance=0;
    dep_deb:Forall(i:nat):s. deposit(i). debit(i)~s;
  Constructor
    new:Self
  Creation
    init_account:new. balance=0;
End Account
```

由上述可确定一个类规范 $\Gamma_{Acc}=\{X_{Acc},F_{Acc},A_{Acc},C_{Acc}\}$，其中 X_{Acc} 表示 Self，函子 $F_{Acc}=(balance:X\to \text{nat}, deposit:X\to X^{\text{nat}}, debit:X\to X^{\text{nat}}, A_{Acc}=\{\text{dep\_bal, deb\_bal, dep\_deb}\}, C=\{\text{init\_account}\}$。

由类规范 Γ_{Acc} 可以给出以下 3 个不同的子类 $C_1=(U_1,\alpha_1,u_0^1)|=\Gamma_{Acc}$、$C_2=(U_2,\alpha_2,u_0^2)|=\Gamma_{Acc}$ 和 $C_3=(U_3,\alpha_3,u_0^3)|=\Gamma_{Acc}$，其定义分别为：

(1) $C_1=\{\text{nat}, balance:\text{nat}\to\text{nat}, deposit:\text{nat}\times\text{nat}\to\text{nat}, debit:\text{nat}\times\text{nat}\to\text{nat}\}$，其中 U_1 为 nat，基调 α_1 定义为：$\forall s,i\in\text{nat}, balance(s)=s, deposit(s,i)=s+i, debit(s,i)=\text{if } s>=i \text{ then } (s-i) \text{ else } 0$。

(2) $C_2=\{\text{List[nat]}, balance:\text{List[nat]}\to\text{nat}, deposit:\text{List[nat]}\times\text{nat}\to\text{List[nat]}, debit:\text{List[nat]}\times\text{nat}\to\text{List[nat]}\}$，其中 U_2 为 List[nat] $=[i_0,i_1,\cdots,i_n]$，$i_0=0$ 表示初始状态，基调 α_2 定义为：

$balance([i_0,i_1,\cdots,i_n])=i_0+i_1+\cdots+i_n$

$deposit([i_0,i_1,\cdots,i_n],i)=[i_0,i_1,\cdots,i_n,i]$

$debit([i_0,i_1,\cdots,i_n],i)=[i_0,i_1,\cdots,i_n,-i]$(if $balance([i_0,i_1,\cdots,i_n])-i>=0$)

$debit([i_0,i_1,\cdots,i_n],i)=[i_0,i_1,\cdots,i_n,-balance([i_0,i_1,\cdots,i_n])]$(otherwise)

(3) $C_3=\{\text{Queue[nat]}, balance:\text{Queue[nat]}\to\text{nat}, deposit:\text{Queue[nat]}\times\text{nat}\to\text{Queue[nat]}, debit:\text{Queue[nat]}\times\text{nat}\to\text{Queue[nat]}\}$，其中 U_3 为 Queue[nat] $=\langle i_0,i_1,\cdots,i_n\rangle$，$i_0=0$ 表示初始状态，基调 α_3 定义为：

$balance(\langle i_0,i_1,\cdots,i_n\rangle)=i_n$

$deposit(\langle i_0,i_1,\cdots,i_n\rangle,i)=\langle i_0,i_1,\cdots,i_n,i_n+i\rangle$

$deposit(\langle i_0,i_1,\cdots,i_n\rangle,i)=\text{if } (i_n-i)>=0 \text{ then } \langle i_0,i_1,\cdots,i_n,i_n-i\rangle \text{ else } \langle i_0,i_1,\cdots,i_n,0\rangle$。

容易证明 $(3,\alpha_1)$ 和 $(1,\alpha_1)$ 是 C_1 的两个对象，$([0,2,1,3],\alpha_2)$ 和 $([0,2,1,-1,-1],\alpha_2)$ 是 C_2 的两个对象，而 $(\langle 0,2,3,1\rangle,\alpha_3)$ 和 $(\langle 0,2,3,2\rangle,\alpha_3)$ 是 C_3 的两个对象。

3 对象行为等价的终结语义

若类规范存在终结共代数，则该终结共代数给出了类规范中函子的最小实现，并提供了该函子的所有可观察行为信息。由于终结共代数的状态就是该状态所能观察到的行为，即其状态与行为是一致的[16]，因此，终结共代数利用从纯观察的角度所得到的行为等价关系给出了对象状态间的互相似关系的一种抽象描述，并可以借助其终结语义及共归纳证明原则证明对象间的行为等价关系。

定义 4 给定类规范 Γ，$(Z,\alpha_Z:Z\to F(Z),z_0)$ 为 Γ 的终结共代数，使得对于任意的类 $(U_1,\alpha_1,u_0^1)|=\Gamma$，都存在唯一的同态射 $beh_{\alpha 1}:U_1\to Z$，称为由 α_1 所确定的共归纳扩展射，使得 $\alpha_Z\circ beh_{\alpha 1}=F(beh_{\alpha 1})\circ\alpha_1$，即满足图 1 所示的图表交换。

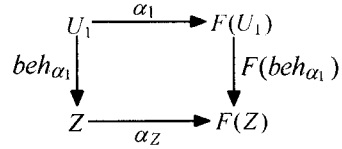

图 1 终结共代数及共归纳扩展射

将 $beh_{\alpha 1}$ 应用于类 C_1 的某个对象 $o_1=(u_1\in U_1,\alpha_1)$ 上，可得到 o_1 从状态 u_1 开始的 α_1 变迁序列的可观察行为 $beh_{\alpha 1}(u_1)$，称之为对象 o_1 在状态 u_1 的全局可观察行为。由终结共代数和共归纳扩展射的性质可知，对任意的对象 $o_1=(u_1\in U_1,\alpha_1)$ 的状态 u_1 以及相应的任意后续状态 u_1'，o_1 所能观察到的行为都等于 (Z,α_Z) 在状态 $beh_{\alpha 1}(u_1)$ 以及它的任意后续状态所能观察到的行为。

由共归纳扩展射的唯一性及终结共代数上的互模拟是等价关系的性质，可得到基于终结共代数的对象行为等价证明方法：两个对象是行为等价，当且仅当它们的对象状态通过共归纳扩展射映射到终结共代数上的同一状态。

定义 5 给定类规范 Γ 及类 $C_1=(U_1,\alpha_1,u_0^1)|=\Gamma$ 和 $C_2=(U_2,\alpha_2,u_0^2)|=\Gamma$，$o_1=(u_1\in U_1,\alpha_1)$ 和 $o_2=(u_2\in U_2,\alpha_2)$ 分别为 C_1 和 C_2 的两个对象。称对象 o_1 和 o_2 是行为等价的，记为 $o_1\approx o_2$，当且仅当 $beh_{\alpha 1}(u_1)=beh_{\alpha 2}(u_2)$，即满足 $u_1\leftrightarrow u_2$。称类 C_1 和 C_2 是行为等价的，记为 $C_1\approx C_2$，当且仅当满足 $beh_{\alpha 1}(U_1)=beh_{\alpha 2}(U_2)$ 和 $beh_{\alpha 1}(u_0^1)=beh_{\alpha 2}(u_0^2)$。

例 2 对于类规范 Γ_{Acc}，类 C_1 是 Γ_{Acc} 的一个终结共代数。因此，可利用 C_1 来证明对象 $([0,2,-1,3],\alpha_2)$ 和 $(\langle 0,2,3,4\rangle,\alpha_3)$ 是行为等价的。即只需要证明共归纳扩展射 $beh_{\alpha 2}$：List[nat]→nat 和 $beh_{\alpha 3}$：Queue[nat]→nat 满足条件 $beh_{\alpha 2}([0,2,-1,3])=beh_{\alpha 3}(\langle 0,2,3,4\rangle)$。

由于对象中方法的各种行为结构通过强 Monads 进行参数化，因此利用终结共代数证明对象的行为等价关系需要考虑强 Monads B 的具体结构。例如，对于 $B=Id$，$B=Id+1$，$B=Id+E$，$B=\boldsymbol{P}_\omega$ 和 $B=Id^*$，$o_1\approx o_2$ 分别定义为：

$B=Id$：$o_1\approx o_2\Leftrightarrow\forall i'\in I'.\ at_Z(beh_{\alpha 1}(u_1),i')=at_Z(beh_{\alpha 2}(u_2),i')\wedge\forall i\in I.\ \exists o\in O.\ ((u_1',o)=op_Z(beh_{\alpha 1}(u_1),i).\ (u_2',o)=op_Z(beh_{\alpha 2}(u_2),i).\ u_1'=u_2')$

$B=Id+1$：$o_1\approx o_2\Leftrightarrow\forall i'\in I'.\ at_Z(beh_{\alpha 1}(u_1),i')=at_Z(beh_{\alpha 2}(u_2),i')\wedge\forall i\in I.\ \exists o\in O.\ ((u_1',o)=op_Z(beh_{\alpha 1}(u_1),i).\ (u_2',o)=op_Z(beh_{\alpha 2}(u_2),i).\ u_1'=u_2'\vee op_Z(beh_{\alpha 1}(u_1),i)=op_Z(beh_{\alpha 2}(u_2),i)=*)$

$B=Id+E$：$o_1\approx o_2\Leftrightarrow\forall i'\in I'.\ at_Z(beh_{\alpha 1}(u_1),i')=at_Z(beh_{\alpha 2}(u_2),i')\wedge\forall i\in I.\ \exists o\in O.\ ((u_1',o)=op_Z(beh_{\alpha 1}(u_1),i).\ (u_2',o)=op_Z(beh_{\alpha 2}(u_2),i).\ u_1'=u_2'\vee\exists e\in E.\ op_Z(beh_{\alpha 1}(u_1),i)=op_Z(beh_{\alpha 2}(u_2),i)=e)$

$B=\boldsymbol{P}_\omega$：$o_1\approx o_2\Leftrightarrow\forall i'\in I'.\ at_Z(beh_{\alpha 1}(u_1),i')=at_Z(beh_{\alpha 2}(u_2),i')\wedge\forall i\in I.\ \exists o\in O.\ (\forall(u_1',o)=op_Z(beh_{\alpha 1}(u_1),i).\ \exists(u_2',o)=op_Z(beh_{\alpha 2}(u_2),i).\ u_1'=u_2'\wedge\forall(u_2',o)=op_Z(beh_{\alpha 2}(u_2),i).\ \exists(u_1',o)=op_Z(beh_{\alpha 1}(u_1),i).\ u_1'=u_2')$

$B=Id^*$：$o_1\approx o_2\Leftrightarrow\forall i'\in I'.\ at_Z(beh_{\alpha 1}(u_1),i')=at_Z(beh_{\alpha 2}(u_2),i')\wedge\forall i\in I.\ \exists o\in O.\ (([u_1',\cdots,u_n'],o)=op_Z$

$(beh_{\alpha 1}(u_1), i).([u_1'', \cdots, u_n''], o) = op_Z(beh_{\alpha 2}(u_2), i). \forall 1 \leqslant j \leqslant n. u_j' = u_j'')$

4 实例与验证

利用 CCSL 给出类规范的描述后，可以通过 CCSL Compiler 将类规范转换成满足 PVS 格式的高阶逻辑，然后在 PVS 理论证明工具中建立各种模型进行验证和理论证明。CCSL 的具体工作环境如图 2 所示。

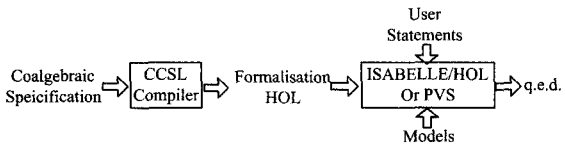

图 2 CCSL 工作环境

以 Γ_{Acc} 为例，利用 CCSL Compiler 对共代数规范文件 Account. beh 进行编译后可得到相应的 PVS 文件 Account_basic. pvs。该文件包含一系列用于描述抽象类 Account 的行为、接口、互相似性和不变量的理论，而 Account 中的各个断言也被转换成 PVS 中相应的谓词。根据 Account_basic. pvs 中所包含的理论就可以建立各种模型来验证规范的一致性以及模型之间的行为等价或精化关系。

要证明对象的行为等价关系，首先必须在 PVS 中建立类的模型，并证明该模型满足类规范。例如，下面的模型 C1 就是 Account 类规范的一个子类。

```
C1: Theory
  Begin
    NatState: Type=nat
    IMPORTING AccountBasic[NatState]
    nat_acc: AccountSignature[NatState]=
    (# balance=Lambda(s: NatState): s,
        deposit=Lambda(s: NatState, i: nat): s+i,
        debit=Lambda(s: NatState, i: nat):
            If s>=i Then s-i Else 0 EndIf #)
    nat_new: AccountConstructors[NatState]=(# new:=0 #)
    nat_assert: Lemma AccountAssert? (nat_acc)
    nat_create: Lemma AccountCreate? (nat_acc)
    nat_model: Proposition AccountModel? (nat_acc, nat_new)
END C1
```

要证明 C1 是 Γ_{Acc} 的一个实现，只需要证明上述 Theory 中的命题 AccountModel? 成立即可。同理，可以建立另外两个模型 C2 和 C3。C2 的状态空间定义为 ListState: Type=list [nat]，基调为 list_acc，初始化变量为 list_new。C3 的状态空间定义为 QueueState: Type=Queue[nat]，基调为 queue_acc，初始化变量为 queue_ new。C1、C2 和 C3 分别给出了例 1 中类 C_1、C_2 和 C_3 在 PVS 中的实现。

对于模型 C1，还可以通过以下命题证明 C1 同时是 Account 类规范的一个终结共代数模型：

C1_Final: Proposition Account_final? (nat_acc)。

在证明 C2 和 C3 中的对象行为等价关系之前，下面先给出 C2 和 C3 上的互模拟关系定义：

C2_C3_Beh_Rel: [[[ListState, QueueState]->bool]->[[ListState, QueueState]->bool]] = Lambda (R: [[ListState, QueueState]->bool]): Lambda (x1: ListState, x2: QueueState): Forall (i: nat): balance(list_acc)(x1) = balance (queue_acc)(x2) and R(deposit(list_acc)(x1, i), deposit (queue_acc)(x2, i)) and R(debit(list_acc)(x1, i), debit(queue_acc)(x2, i));

C2_C3_Beh_bisimulation?: [[[ListState, QueueState]->bool]->bool] = Lambda(R: [[ListState, QueueState]->bool]): Forall(x1: ListState, x2: QueueState): R(x1, x2) Implies C2_C3_Beh_Rel(R)(x1, x2);

C2_C3_Beh_bisim?: [[ListState, QueueState]->bool] = Lambda (x1: ListState, x2: QueueState): Exists (R: [[ListState, QueueState]->bool]): C2_C3_Beh_bisimulation? (R) and R(x1, x2);

其中，C2_C3_Beh_Rel 给出了 C2 和 C3 的状态空间上的一个二元关系，且该二元关系在所有方法下保持。而 C2_C3_Beh_bisimulation? 则给出了 C2 和 C3 间的一个互模拟关系。C2_C3_Beh_bisim? 说明了 C2 和 C3 中的两个状态是互相似的，当且仅当存在一个包含这两个状态的互模拟关系。

要利用 C1 证明对象 $o_1 = ([0, 2, -1, 3], \alpha_2)$ 和 $o_2 = (\langle 0, 2, 3, 4\rangle, \alpha_3)$ 是行为等价的，只需证明若 o_1 和 o_2 的对象状态通过共归纳扩展射映射到 C1 中的同一状态，则存在一个包含这两个对象状态的互模拟关系。因此，首先建立 C2 和 C3 的状态空间到 C1 的映射 ListMorp 和 QueueMorp，并证明它们确实是共归纳扩展射。

ListMorp?: [[ListState -> NatState] -> bool] = Lambda(f: [ListState -> NatState]): Forall(x: ListState, i: nat): f(x) = car(x) and balance(list_acc)(x) = balance(nat_acc)(f(x)) and f(deposit(list_ acc)(x, i)) = deposit(nat_acc)(f(x), i) and f(debit(list_acc)(x, i)) = debit(nat_acc)(f(x), i);

ListMorp_ uni?: bool = Forall (f1: [ListState -> NatState], f2: [ListState -> NatState]): ListMorp? (f1) and ListMorp? (f2) implies f1 = f2;

QueueMorp?: [[QueueState -> NatState] -> bool] = Lambda(g: [QueueState -> NatState]): Forall(x: QueueState, i: nat): g(x) = top(x) and balance(queue_acc)(x) = balance (nat_acc)(g(x)) and g(deposit(queue_acc)(x, i)) = deposit(nat_acc)(g(x), i) and g(debit(queue_ acc)(x, i)) = debit(nat_acc)(g(x), i);

QueueMorp_ uni?: bool = Forall (g1: [QueueState -> NatState], g2: [QueueState -> NatState]): QueueMorp? (g1) and QueueMorp? (g2) Implies g1 = g2;

is_object_final_equal: Proposition Forall(f: [ListState -> NatStat e], g: [QueueState -> NatState]): ListMorp? (f) and ListMorp _uni? and QueueMorp? (g) and QueueMorp_uni? and f(cons(3, cons(-1, cons(2, cons(0, null))))) = g(put(4, put(3, put(2, put(0))))) Implies C2_C3_Beh_bisim? (cons(3, cons(-1, cons(2, cons(0, null)))), put(4, put(3, put(2, put(0))))));

利用 PVS 可证明上述命题 is_object_final_equal 确实成立。因此，对象 o_1 和 o_2 是行为等价的。

结束语 作为代数中初始代数的范畴对偶概念，终结共代数给出了一种以单步转换为基础的共递归定义的行为等价

(下转第 190 页)

关，始终为$|W|=10000$。而GMV随着$|U|$的增加，需要的计算次数单调减少，这是由于不确定数据集规模增大，网格顶点的top-k概率均会变小，其reverse top-k的结果集合也相应变小，从而GMV需要进行的概率计算次数减少。随着$|U|$从2k增加到10k，GMV需要的概率计算次数分别为Naive的17.4%，9.25%，7.83%，4.72%，4.1%，说明U的规模越大，GMV优势越明显。

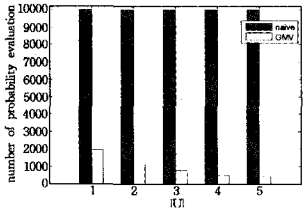

图4 GMV相对于$|U|$的扩展性

结束语 本文研究了不确定数据上的reverse top-k查询，提出了基于物化视图的查询优化算法GMV。实验结果表明，GMV算法对数据集规模有较好的可扩展性。

参考文献

[1] Ilyas I F, Beskales G, Soliman M A. A survey of top-k query processing techniques in relational database systems[J]. ACM Computer Survey, 2008, 40(4): 1-58

[2] Vlachou A, Doulkeridis C, Kotidis Y, et al. Reverse Top-k Queries[C]// Proc. of ICDE. 2010: 365-376

[3] Abiteboul S, Kanellakis P, Grahne G. On the Representation and Querying of Sets of Possible Worlds[J]. Theoretical Computer Science, 1991, 78(1): 34-48

[4] Borzsony S, Kossmann D, Stocker K. The skyline operator[C]// Proc. of ICDE. 2001: 421-430

[5] Lian X, Chen L. Monochromatic and bichromatic reverse skyline search over uncertain databases[C]// Proc. of SIGMOD. 2008: 213-226

[6] Wu X, Tao Y, Wong R C, et al. Finding the Influence Set through Skylines[C]// Proc. of EDBT. 2009: 1030-1041

[7] Aggarwal C C, Yu P S. A Survey of Uncertain Data Algorithms and Applications[J]. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(5): 609-623

[8] 周傲英，金澈清，王国仁，等. 不确定性数据管理技术研究综述[J]. 计算机学报，2009，32(1)：1-16

[9] Soliman M A, Ilyas I F, Chang K C-C. Top-k query processing in uncertain databases[C]// Proc. of ICDE. 2007: 896-905

[10] Hua M, Pei J, Zhang W, et al. Ranking Queries on Uncertain Data: A Probabilistic Threshold Approach[C]// Proc. of SIGMOD. 2008: 673-686

[11] Zhang X, Chomicki J. On the Semantics and Evaluation of Top-k Queries in Probabilistic Databases[C]// Proc. of DBRank. 2008: 556-563

[12] Cormode G, Li F, Yi K. Semantics of Ranking Queries for Probablistic Data and Expected Ranks[C]// Proc. of ICDE. 2009: 305-316

[13] Li J, Saha B, Deshpande A. A Unified Approach to Ranking in Probabilistic Databases[C]// Proc. of VLDB. 2009: 502-513

(上接第182页)

证明方法，即通过共归纳扩展射将两个不同的元素映射到终结共代数中的同一状态上。这不仅可以降低系统对行为等价关系的证明难度，也便于借助形式化验证工具进行证明。而强Monads可进一步提高共代数在面向对象方法的形式化描述中的抽象性。

在下一步的工作中，我们将继续探讨不同类规范下的对象行为等价关系及其行为精化语义。

参考文献

[1] Papathomas M. Language Design Rationale and Semantic Framework for Concurrent Object-oriented Programming [D]. Dept. of Computer Science, University of Geneva, 1992

[2] Hameurlain N, Sibertin-Blanc C. Behavioural Types in CoOperative Objects [C]// Moreira A M D, Demeyer S, eds. Object-Oriented Technology, ECOOP 99 Workshop Reader, ECOOP 99 Workshops, Panels, and Posters, Lisbon. Volume 1743 of Lecture Notes in Computer Science, Springer, 1999: 320-321

[3] Peter C, Puntigam F. A Concurrent Object Calculus with Types that Express Sequences [C]// Proceedings of the ECOOP Workshop on Semantics of Objects as Processes(SOAP'99). Lisbon, Portugal, 1999: 321

[4] Goel A, Roychoudhury A, Thiagarajan P S. Interacting Process Classes [J]. ACM Transactions on Software Engineering Methodology, 2009, 18(4): 1-47

[5] Burstall R, Diaconescu R. Hiding and Behaviour: an Institutional Approach [M]// Roscoe A W, ed. A Classical Mind. Essays in honour of C. A. R. Hoare, 1994: 75-92

[6] Berrima M, Rajeb N B. Linking Algebraic Observational Equivalence and Bisimulation [C]// Lecture Notes in Computer Science, Volume 6224. 2010: 76-87

[7] Rutten J. Universal Coalgebra: a Theory of Systems [J]. Theoretical Computer Science, 2000: 3-80

[8] 周晓聪，舒忠梅. 计算机科学中的共代数方法的研究综述[J]. 软件学报，2003，4(10)：1661-1671

[9] Reichel H. An Approach to Object Semantics Based on Terminal Coalgebras [J]. Mathematical Structures in Computer Science, 1995, 5(2): 129-152

[10] Jacobs B. Objects and Classes, Co-algebraically [M]// Freitag B, Jones C B, Lengauer C, eds. Object-orientation with Parallelism and Persistence Kluwer Acad. Publ., 1996: 83-103

[11] Jacobs B, Rutten J. Inheritance and Cofree Construction [C]// Cointe P, ed. European Conference on Object-oriented Programming, Lecture Notes in Computer Science, Vol. 1098. Springer, Berlin, 1996: 210-231

[12] Jacobs B. Invariants, Bisimulations and the Correctness of Coalgebraic Refinements [C]// Proceedings of the 6th International Conference on Algebraic Methodology and Software Technology. Lecture Notes In Computer Science, Vol. 1349. 1997: 276-291

[13] Jacobs B. Introduction to Coalgebra: Towards Mathematics of States and Observations [M]. Book Draft. http://www.cs.ru.nl/~bart, 2005

[14] Moggi E. Notions of Computation and Monads [J]. Inf. & Comp., 1991, 93(1): 55-92

[15] Tews H. The Coalgebraic Class Specification Language CCSL-Syntax and Semantics [R]. TU Dresden technical report TUD-FI02-08-August. 2002

[16] Jacobs B, Rutten J. A Tutorial on(Co) Algebras and(Co) Induction [J]. EATCS Bulletin, 1997, 62: 222-259