计算机科学中的共代数方法的研究综述*

周晓聪+, 舒忠梅

(中山大学 计算机科学系,广东 广州 510275)

A Survey on the Coalgebraic Methods in Computer Science

ZHOU Xiao-Cong+, SHU Zhong-Mei

(Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, China)

+ Corresponding author: Phn: 86-20-84034709, E-mail: isszxc@zsu.edu.cn

http://www.cs.zsu.edu.cn/jzyg/zxc/zxc.htm

Received 2003-04-18; Accepted 2003-06-04

Zhou XC, Shu ZM. A survey on the coalgebraic methods in computer science. *Journal of Software*, 2003, 14(10):1661~1671.

http://www.jos.org.cn/1000-9825/14/1661.htm

Abstract: Unlike the widespread applications of algebraic methods in computer science, coalgebraic methods, as the dual concepts of algebras, have not been noticed by computer scientists till the mid 1990s. In algebraic methods, the constructive elements of data types are studied, while in coalgebraic methods, the observable behaviors of systems are investigated. Coalgebraic methods have distinct advantages in mathematic study of state-based systems since them enable the depth research on those systems' properties such as behavior equivalence, nondeterminism and so on. Coalgebraic methods have also been applied in many research fields, for example, automata theory, semantics of concurrency, specifications of object-oriented software etc. The recent progress of coalgebraic methods, including its basic concepts, categorical foundations, logical foundations and its applications, is summarized for raising the attention of the relative researchers.

Key words: coalgebra; bisimulation; final coalgebra; coinduction; coalgebraic logic

摘 要: 代数理论已经在抽象数据类型、程序语义等计算机科学领域有了广泛的应用,而代数的对偶概念——共代数,则直到20世纪90年代中后期才被越来越多的计算机学者关注.代数从"构造"的角度研究数据类型,而共代数则从"观察"的角度考察系统及其性质.共代数方法对研究基于状态的系统有独特的优越性,可以对系统的行为等价、不确定性等从数学上进行深入的探讨.目前,共代数理论已经逐步应用在自动机理论、并发程序的语义、面向对象程序的规范等领域.对共代数的基本概念、范畴理论基础、共代数逻辑及应用等方面的最新研究成果进行了介绍,以引起国内相关研究领域的学者对计算机科学中的共代数方法的关注.

关键词: 共代数;互模拟;终结共代数;共归纳原理;共代数逻辑

中图法分类号: TP301 **文献标识码**: A

* **第一作者简介**: **周晓聪**(1971－),男,湖南常宁人,博士,副教授,主要研究领域为形式语义学,面向对象技术的形式理论基础,软件体系结构.

共代数(coalgebra)是代数(algebra)的对偶(dual)概念.在计算机科学中,代数方法从“构造(constructive)”的角度研究抽象数据类型的语义,而共代数方法则从“观察(observable)”的角度描述诸如对象、自动机、进程、软件构件等基于状态的系统.

例如,从代数角度来看,(有穷)表 LIST 是通过给出空链表(nil:1→LIST)及往表中增加类型 A 的元素(cons: LIST×A→LIST)这两个操作构造而得,可使用归纳原理定义链表的操作并证明其有关性质;对应地,从共代数角度来看,无穷表(流)STREAM 可进行两种最基本的“观察”:查看它的首元素(head:STREAM→A)和查看它的其余元素组成的表(tail:STREAM→STREAM),通过这两个基本行为,使用共归纳(coinduction)原理可定义无穷表的更多行为,并证明其有关性质.

在计算机科学中,代数方法已经在抽象数据类型、程序的形式语义等领域有了广泛的应用,但共代数方法近年来才引起计算机学者的注意与研究.1988 年,P. Aczel 等人首次使用共代数方法给出了 CCS 中进程代数(process algebra)的终结语义(final semantic)[1].此后,计算机科学工作者发现共代数方法对研究基于状态的系统(例如自动机、进程、对象、软件构件等)有独特的优越性.使用共代数理论作为工具,可以对系统的行为等价、不确定性等从数学上进行深入探讨,为这些系统建立良好的形式理论基础,并对其性质进行描述与验证.

20 世纪 90 年代初,J. Rutten 等人将共代数方法用于自动机理论、并发程序语义的研究[2~4],并对其基本概念和数学基础进行了深入探讨,奠定了泛共代数理论(universal coalgebra theory)的基础[5,6].另一方面,H. Reichel, B. Jacobs 等人将共代数方法用于面向对象程序的规范与语义的研究[7~11],开发了 LOOP(logic of object-oriented programming)工具对 OO 规范进行描述与验证[12,13].B. Jacobs 等人还在归纳与共归纳的范畴论基础[14]、共代数与模态逻辑的关系等方面进行了深入的研究[15~17].

90 年代中后期以来,越来越多的计算机学者对共代数方法进行了研究,如 J. Adámek, H. P. Gumm 等人进一步研究了它的范畴论基础[18~22],L. Moss, A. Kurz, D. Pattinson 等人进一步研究了共代数逻辑[23~28],G. Roşu, J. Rothe 等人进一步研究了共代数规范[29~32]等等.自 1998 年以来,国际上已经召开了 5 届有关计算机科学中的共代数方法(Coalgebraic Methods in Computer Science, CMCS’98~ CMCS 2002)的年会.目前,这一领域的研究正在不断发展,上面列出的只是其中具有代表性的工作.

我们发现,对于计算机科学中的共代数方法,当前研究主要集中在共代数方法的范畴论基础、逻辑基础及应用这 3 个方面.本文就这几方面对这一领域的研究工作进行综述.第 1 节介绍共代数的定义,并给出几个典型例子加以说明;第 2 节综述共代数方法在范畴论基础方面的研究;第 3 节介绍共代数逻辑方面的研究工作;第 4 节概述共代数方法在计算机科学中的应用;第 5 节总结全文,并展望共代数方法的发展.由于受篇幅的限制,本文内容不可能囊括这一领域的所有研究成果,只能概述其主要研究工作及成果,但希望能引起相关研究领域的读者,特别是研究并发程序语义、面向对象的形式理论基础等领域的读者对共代数方法的兴趣与关注.

1 共代数的定义

众所周知,代数是一个集合 A 及其上的运算集$\Omega=\{f:A^n\to A\}$.运算 $f:A^n\to A$ 的直接对偶形式是 $f':A\to A+A+...+A$,但这难以带来有意义的结果,因此,共代数作为代数的对偶概念,最初并没有引起人们太多的注意.在范畴理论中,给定自函子 $F:\mathbf{C}\to\mathbf{C}$,代数被抽象定义为二元组$\langle A,f:FA\to A\rangle$,值得研究的共代数概念是在该意义下的对偶概念.

定义 1. 给定范畴 $\mathbf{C}$ 和自函子 $F:\mathbf{C}\to\mathbf{C}$,$F$-共代数是二元组$\langle A,\alpha\rangle$,其中 A 是 $\mathbf{C}$ 的对象;$\alpha:A\to FA$ 是 $\mathbf{C}$ 的射,称为 A 上的转换结构(transition structure).F-共代数$\langle A,\alpha\rangle$到$\langle B,\beta\rangle$的 F-态射(homomorphism)是 $\mathbf{C}$ 的射 $h:A\to B$,且满足 $Fh\circ\alpha=\beta\circ h$.不难验证,$F$-共代数和 F-态射构成了一个范畴,记为 $\mathbf{C}_F$.

直观地来看,F-共代数$\langle A,\alpha\rangle$是一个具有内部状态的系统,A 是其所有可能的状态,射$\alpha:A\to FA$ 代表从外部可观察的系统行为,这种行为可能会影响系统内部的状态.α往往可分解为几个射,每个射代表系统可观察的一个行为或系统对外部观察的一种响应.F-态射则是保持系统行为的射.下面的例子说明了共代数与基于状态的系统之间的密切联系.

例 1(流/无穷序列):固定集合 A,定义函子 $St_A:\mathbf{Set}\to\mathbf{Set}$ 为 $St_A(h:X\to Y)=id_A\times h:A\times X\to A\times Y$.

对于 St_A-共代数$\langle X,\alpha:X\to A\times X\rangle$,$X$ 是状态集,射α可看作$\langle value,next\rangle:X\to A\times X$,其中 $value:X\to A$ 表示从 X 中观察值 $a\in A$,$next:X\to X$ 表示 X 转入下一状态,以便进行下一次观察.也可将 X 看成输入/输出流(stream)$\{a_0,a_1,a_2,...,a_n,...\}$,$value:X\to A$ 取得流的第 1 个输入/输出值,$next:X\to X$ 表示流准备好输入/输出下一个值.

对于两个流$\langle X,\alpha:X\to A\times X\rangle$和$\langle Y,\beta:Y\to A\times Y\rangle$,函数 $h:X\to Y$ 是 St_A-态射当且仅当$\forall x\in X,value_Y(h(x))=value_X(x)$和$\forall x\in X,next_Y(h(x))=h(next_X(x))$.

在程序设计语言的形式语义研究中,由于程序的输入/输出是一个动态过程,传统的以域论为基础的研究方法很难描述输入/输出的形式语义,上例则表明,基于共代数的方法可以从数学上深入探讨输入/输出流的性质,很好地给出它们的形式语义描述,可参考 B. Jacobs 和 J. Rutten 的讲稿[33],他们给出了这方面的一些很好的例子.进一步地,U. Hensel 在文献[34]中详细讨论了序列/流的性质,并比较了在描述序列/流的形式语义时,基于共代数方法和基于域论方法的优缺点.

例 2(带标号转换系统):固定集合 A,定义函子 $Lt_A:\mathbf{Set}\to\mathbf{Set}$ 为 $Lt_A(h:X\to Y)=\mathcal{P}(id_A\times h):\mathcal{P}(A\times X)\to\mathcal{P}(A\times Y)$.

Lt_A-共代数$\langle X,\alpha:X\to\mathcal{P}(A\times X)\rangle$可看作带标号的转换系统,射$\alpha$是转换函数,$X$ 是状态集合,A 是标号(动作)集,$\mathcal{P}(A\times X)$是 $A\times X$ 的幂集.对于当前状态 $x\in X$,x 转换到状态 $x'\in X$ 并输出 $a\in A$ 当且仅当$\langle a,x'\rangle\in\alpha(x)$.

对于两个 Lt_A-共代数$\langle X,\alpha:X\to\mathcal{P}(A\times X)\rangle$和$\langle Y,\beta:Y\to\mathcal{P}(A\times X)\rangle$,不难验证函数 $h:X\to Y$ 是 Lt_A-态射当且仅当$\forall x\in X$,有$\forall x'\in X,\langle a,x'\rangle\in\alpha(x)\Rightarrow\langle a,h(x')\rangle\in\beta(h(x))$和$\forall y\in Y,\langle a,y\rangle\in\beta(h(x))\Rightarrow\exists x'\in X(y=h(x')\wedge\langle a,x'\rangle\in\alpha(x))$成立.

传统的自动机理论以自动机之间的态射和互模拟为其核心概念,而这两个概念也正是共代数方法的核心.上例表明,自动机与共代数有着密切的联系,但从共代数的角度去研究自动机则进一步拓广了自动机理论的研究内容,例如,终结自动机的概念将成为自动机理论的核心概念之一.J. Rutten 使用共代数方法对自动机理论进行了初步的探讨,获得了一些新的结果,表明共代数方法在这一领域具有独特的优势[3].

例 3(面向对象类规范):假定有下述类规范:

```
ClassSpec ACC {                    // 银行帐户
    deposit: ACC×R→ACC;            // 往帐户存金额,R 为实数集.
    balance: ACC→R;                // 取帐户余额
}
```

这样可以确定函子 $Acc:\mathbf{Set}\to\mathbf{Set}$ 为 $Acc(h:X\to Y)=(id_R\Rightarrow h)\times id_R:X^R\times R\to Y^R\times R$.

对于 Acc-共代数$\langle X,\alpha:X\to X_R\times R\rangle$,$X$ 可看成 ACC 具体实现时的内部表示,射α表示它的公有方法$\langle deposit,balance\rangle:X\to X^R\times R$,$deposit:X\to X^R$ 等价于 $deposit:X\times R\to X$,而 $balance:X\to R$.

对于该规范,取 $X=R$,Acc-共代数$\langle R,\langle deposit_R,balance_R\rangle:R\to R^R\times R\rangle$意味着用一个实数表示该帐户的内部状态(即帐户余额),这时可定义 $deposit_R:R\times R\to R$ 为实数加法,而 $balance_R:R\to R$ 为实数上的恒等函数.

该规范的另一实现可取 $X=R^*$,Acc-共代数$\langle R^*,\langle deposit_{R^*},balance_{R^*}\rangle:R^*\to R^{*R}\times R\rangle$意味着用一个实数序列表示该帐户的内部状态(即记录帐户存钱的过程),$deposit_{R^*}:R^*\times R\to R$ 可定义为$\forall\langle x_1,...,x_n\rangle\in R^*,x\in R$,$deposit_{R^*}(\langle x_1,...,x_n\rangle,x)=\langle x_1,...,x_n,x\rangle$,而 $balance_R:R^*\to R$ 可定义为$\forall\langle x_1,...,x_n\rangle\in R^*,balance_R(\langle x_1,...,x_n\rangle)=x_1+...+x_n$.

对于两个 Acc-共代数$\langle X,\langle deposit_X,balance_X\rangle:X\to X^R\times R\rangle$和$\langle Y,\langle deposit_Y,balance_Y\rangle:Y\to Y^R\times R\rangle$,不难验证函数 $h:X\to Y$ 是 Acc-态射当且仅当$\forall x\in X,balance_X(x)=balance_Y(h(x))$和$\forall r\in R,h(deposit_X(x,r))=deposit_Y(h(x),r)$成立.

对于上述两个 Acc-共代数$\langle R,\langle deposit_R,balance_R\rangle:R\to R^R\times R\rangle$和$\langle R^*,\langle deposit_{R^*},balance_{R^*}\rangle:R^*\to R^{*R}\times R\rangle$,可定义函数 $h:R^*\to R$ 为$\forall\langle x_1,...,x_n\rangle\in R^*,h(\langle x_1,...,x_n\rangle)=x_1+...+x_n$,则不难验证 h 是 Acc-态射.

面向对象技术虽然已广泛使用,但其理论基础的研究仍十分薄弱.上例表明,共代数方法作为研究面向对象技术理论基础的数学工具十分合适,不仅可用于描述和验证对象的性质,而且还可用于描述对象与对象之间的联系.B. Jacobs 等人使用共代数方法研究了面向对象技术的理论基础,特别地,对 Java 语言的语义及 Java 程序的验证进行了深入的探讨[8~13,35,36].

2 共代数的范畴论基础研究

这里讨论的共代数方法的基础研究不是纯粹数学意义上(如在 Hopf 代数中)的研究,而是从计算机科学出

发,为了使共代数方法可以作为研究各种基于状态的软件系统的数学工具而进行的基础研究.这种研究本身又以范畴理论作为工具,因为范畴理论是一种通用的、抽象的,研究各种数学结构的数学语言,而共代数正是一种典型的数学结构.我们将这种基础研究归纳为以下几个方面:

2.1 共代数的构造

代数上的基本构造包括子代数(subalgebra)、积(product)代数与商(quotient)代数,对应地,对于共代数的构造,人们主要研究子共代数(subcoalgebra)、共代数的共积(coproduct)以及商共代数的性质与结构,研究它们与共代数态射的关系.

通过对比,J. Rutten 等人得到了与泛代数理论相对应的一些关于共代数构造的结论[5].比如,子共代数的并是子共代数,所有的子共代数构成了一个完备格,商共代数可看作满同态的同态像等.这些结论都基于集合范畴的一个重要性质:每个函数都可以在同构的意义下惟一地分解为一个单射函数和满射函数的复合.在范畴理论中,将集合范畴中所有单射构成的集合和所有满射组成的集合一起称为集合范畴的分解系统.

为研究在一般范畴中的共代数,需要将分解系统推广到一般范畴,研究这种分解系统要满足怎样的性质才能保证其上的共代数范畴具有良好的结构.目前这方面的主要成果是 A. Kurz 给出的基础范畴和函子要满足的一些公理[24],这些公理使得那些基于集合范畴的重要性质可以推广到一般范畴上,不过有一些重要的理论基础工作仍在进一步研究中.值得指出的是,在任意范畴上讨论共代数范畴将很有意义.例如,在 Scott 域范畴上讨论共代数范畴对讨论并发程序的语义就有重要意义.

2.2 共代数之间的互模拟关系

共代数方法的核心概念之一是互模拟(bisimulation)关系,它从数学上刻画了系统行为等价这个直观概念.系统行为等价本身是一个直观描述,是指两个系统(如对象、进程等)从观察者的角度看可以互相模拟对方的行为,即从观察者开始观察起,两个系统对同样的输入/通信会产生相同的输出/响应,注意,观察者不能观察到(或说区分)系统内部状态的不同.

为了讨论这方面的研究内容,下面我们先给出共代数方法中对互模拟关系的定义和例子:

定义 2. 给定函子 F:$\mathbf{Set}\to\mathbf{Set}$ 及 F-共代数$\langle A,\alpha\rangle$和$\langle B,\beta\rangle$,关系 $\mathrm{R}\subseteq A\times B$ 是$\langle A,\alpha\rangle$和$\langle B,\beta\rangle$之间的(F-)互模拟关系,如果存在函数γ:$\mathrm{R}\to F\mathrm{R}$ 使得 $F\pi_A\circ\gamma=\alpha\circ\pi_A$ 和 $F\pi_B\circ\gamma=\beta\circ\pi_B$ 成立,这里的π_A 和π_B 定义为$\forall\langle a,b\rangle\in\mathrm{R},\pi_A(\langle a,b\rangle)=a$,$\pi_A(\langle a,b\rangle)=b$.

例 4:固定集合 A,对于前面所定义的函子 St_A:$\mathbf{Set}\to\mathbf{Set}$,$\mathrm{R}\subseteq X\times Y$ 是两个流$\langle X,\langle value_X,next_X\rangle:X\to A\times X\rangle$和$\langle Y,\langle value_Y,next_Y\rangle:Y\to A\times Y\rangle$之间的互模拟关系当且仅当$\forall\langle x,y\rangle\in\mathrm{R},value_X(x)=value_Y(y)$且$\langle next_X(x),next_Y(y)\rangle\in\mathrm{R}$ 成立.这表明两个流 X 和 Y 是互模拟的,当且仅当从观察点(流的某个内部状态)$x\in X$ 和 $y\in Y$ 起,两个流的输出/输入相同($value_X(x)=value_Y(y)$),而且对于任何后续状态,它们的输出/输入也将相同(因为$\langle next_X(x),next_Y(y)\rangle\in\mathrm{R}$),这符合直观意义上的系统行为等价.

例 5:固定集合 A,对于前面所定义的函子 Lt_A:$\mathbf{Set}\to\mathbf{Set}$,$\mathrm{R}\subseteq X\times Y$ 是 Lt_A-共代数$\langle X,\alpha:X\to\mathcal{P}(A\times X)\rangle$和$\langle Y,\beta:Y\to\mathcal{P}(A\times X)\rangle$之间的互模拟关系,当且仅当$\forall\langle x,y\rangle\in\mathrm{R}$ 有:

(1) $\forall x'\in X,\langle a,x'\rangle\in\alpha(x)\Rightarrow\exists y'\in Y(\langle x',y'\rangle\in\mathrm{R}\wedge\langle a,y'\rangle\in\beta(y))$;

(2) $\forall y'\in Y,\langle a,y'\rangle\in\beta(y)\Rightarrow\exists x'\in X(\langle x',y'\rangle\in\mathrm{R}\wedge\langle a,x'\rangle\rangle\in\alpha(x))$.

上述条件的直观含义是,两个系统 X 和 Y 是互模拟的,当且仅当从观察点 $x\in X$ 和 $y\in Y$ 起它们是互模拟的,且无论它们分别转换到哪个后续状态,这两个后续状态也都是互模拟的,这与进程代数中互模拟的概念一致.

例 6:对于前面所定义的函子 Acc:$\mathbf{Set}\to\mathbf{Set}$,$T\subseteq X\times Y$ 是 Acc-共代数$\langle X,\langle deposit_X,balance_X\rangle:X\to X^R\times R\rangle$和$\langle Y,\langle deposit_Y,balance_Y\rangle:Y\to Y^R\times R\rangle$之间的互模拟关系当且仅当$\forall\langle x,y\rangle\in T,balance_X(x)=balance_Y(y)$ 和$\langle deposit_X(x),deposit_Y(y)\rangle\in T$ 成立,这也符合系统行为等价的直观含义,即当前观察点两者的存款余额相同,而且以后只要存款额相同,则两者的余额总相等.对于前面定义的两个 Acc-共代数$\langle R,\langle deposit_R,balance_R\rangle:R\to R^R\times R\rangle$和$\langle R^*,\langle deposit_{R^*},balance_{R^*}\rangle:R^*\to R^{*R}\times R\rangle$,定义关系 $T\subseteq R\times R^*$为

$$T=\{\langle x,\langle x_1,x_2,\ldots,x_n\rangle\rangle\mid x=x_1+x_2+\ldots+x_n\},$$

则 T 是它们之间的互模拟关系.

由于互模拟在并发模型,特别是在进程代数中所具有的重要性,J. Rutten[5],A. Kurz[24],H. P. Gumm[20]等人对基于共代数的互模拟关系进行了广泛而深入的研究.例如,其中一个最重要的研究结果是两个共代数之间的互模拟关系构成一个完备格.此外,他们对互模拟关系与共代数同态、子共代数、商共代数等之间的关系也进行了深入研究.

A. Kurz 在其博士论文[24]中指出,最大互模拟关系的转换结构(即上述定义 2 中的函数γ :R→FR)可能不是惟一的(至少对于幂集函子 $\mathcal{P}$ 是这样),所以在讨论最大互模拟关系时应不考虑其转换结构的比较.同时,他给出了使得最大互模拟关系的转换结构惟一的条件.关于最大互模拟关系,A. Kurz 提出了以下问题[24]:

(1) 是否最大互模拟关系存在某种意义上的最大转换结构?这种最大转换结构可能对系统间互模拟关系的实现有指导意义.

(2) 是否可能给出某些函子具有惟一转换结构的最大互模拟关系?是否可能给出某些函子的最大互模拟关系等于笛卡尔积的函子?

(3) 是否可利用具有惟一转换结构的互模拟关系来定义所谓确定型共代数函子?这种确定性与传统的确定性有什么关系?

从目前的文献来看,H.P. Gumm 在文献[37]中解决了上述问题(2),但问题(1)和问题(3)尚没有给出合适的回答.我们认为问题(3)的解决有助于对计算机科学中广泛使用的确定性和非确定性有更深入的认识.A. Kurz 在其博士论文[24]中还试图在一般范畴中定义描述行为等价的其他概念——共同余关系(cocongruence)和观察等价,并研究了它们与互模拟之间的对应关系,这为研究系统行为等价性提供了一种新思路.

2.3 终结共代数的存在性

在代数理论中经常使用归纳定义和归纳证明.归纳定义的基础是初始(initial)代数的性质,即初始代数到任意代数存在惟一的同态射,因此适当地构造一些代数就(归纳)定义了初始代数到该代数的惟一同态射.进一步地,可以定义最小(minimal)代数.最小代数没有真子代数,从而可以对定义在最小代数上的一些谓词进行归纳证明,即只要证明满足该谓词的元素组成的集合是其子代数,即可得到最小代数的所有元素满足该谓词.

归纳原理和归纳证明是将代数理论应用到计算机科学,特别是抽象数据类型理论研究的重要基础.对应地,共归纳定义和共归纳证明是将共代数方法用于系统的规范描述等领域的重要理论基础.共归纳的基础是终结(final/terminal)共代数的性质:任意共代数到终结共代数存在惟一的同态射,这使得终结共代数的最大互模拟关系就是恒等关系,从而要证明终结共代数中两个元素相等,只要证明它们是互模拟的即可.进一步地,类似于最小代数的概念,可引入简单共代数,要证明简单共代数中两个元素相等,只要证明它们是互模拟的即可.关于共归纳原理的应用例子,参见 B. Jacobs 和 J. Rutten 的讲稿[33].

由于共归纳原理的重要性,终结共代数的存在条件就成为共代数方法的基础研究中的一个重要方面.J. Rutten 等人已证明[5],像上述例 1 中的函子 St_A 和例 3 中的函子 Acc 这样的多项式自函子(由恒等函子、常函子通过积、共积构造得到的函子)都存在终结共代数,但像上述例 2 中的函子 Lt_A:$\mathbf{Set}\to\mathbf{Set}$,由于 $Lt_A(X)=\mathcal{P}(A\times X)$,而 X 不可能与 $\mathcal{P}(X)$等势,更不可能与 $\mathcal{P}(A\times X)$等势,所以由 Lambek 引理(该引理断定共代数上的转换结构是同构射),函子 Lt_A 不存在终结共代数.但若约定系统的任一状态的下一可能状态组成有限集合,则得到函子 Lt_{Af}:$\mathbf{Set}\to\mathbf{Set}$,定义为:对任意函数 $f:X\to Y$,$Lt_{Af}(f)=\mathcal{P}_f(id_A\times f):\mathcal{P}_f(A\times X)\to\mathcal{P}_f(A\times Y)$,其中 $\mathcal{P}_f(A\times X)$表示 $A\times X$ 的所有有限子集构成的集合;$\mathcal{P}_f(id_A\times f)$定义与 $\mathcal{P}(id_A\times f)$的定义相同,但只作用在有限集上,则函子 Lt_{Af}存在终结共代数.

更一般地,H.P. Gumm 等人证明了若函子 F:$\mathbf{Set}\to\mathbf{Set}$ 是有界函子,则它存在终结共代数[20~22],这里的有界是指存在基数κ使得对任意 F-共代数$\langle A,\alpha\rangle$及$\forall a\in A$,都存在$\langle A,\alpha\rangle$的子共代数$\langle B,\beta\rangle$满足 $a\in B$ 且$|B|\le\kappa$.为了将共代数理论建立在一般范畴上,A. Kurz 将有界函子概念推广到有界范畴(bounded category),证明了一般范畴下的终结共代数存在定理[24].

3 共代数逻辑的研究

与代数类似,给定函子 $F:\mathcal{C}\to\mathcal{C}$,给出 F-共代数$\langle A,\alpha\rangle$只是给出其基调(signature),必须对α中所给出的行为进行约束才能描述系统的性质.例如,对于函子 $Acc:\mathbf{Set}\to\mathbf{Set}$,Acc-共代数$\langle X,\langle deposit_X,balance_X\rangle:X\to X^R\times R\rangle$只是给出该类的对象必须有 *deposit* 和 *balance* 两个操作,这还不是一个完整的类规范,必须描述这两个操作所满足的性质.

1997年,L. Moss最早在文献[25]中研究适合描述共代数性质的逻辑,指出共代数与模态逻辑有密切关系.A. Kurz 在文献[38]中提出使用模态逻辑描述基于共代数方法的面向对象类规范中的约束.近 5 年来,许多学者对适用于共代数的逻辑、模态逻辑和共代数之间的关系以及与代数理论中的 Birkhoff 代数簇定理对应的 Birkhoff 共代数簇定理进行了深入研究.A. Kurz 总结了共代数逻辑研究的 3 种主要途径[23]:

(1) L. Moss[25]提出了共代数逻辑方法.他针对保持弱回拉的函子 $F:\mathbf{Set}\to\mathbf{Set}$,提供一种通用的办法,给出用于描述共代数性质的逻辑语言.这种从函子产生逻辑的方法十分通用,但语法和语义形式复杂,与模态逻辑的关系不明显,因此难以直接应用.

(2) M. Röβiger[39]和 B. Jacobs[15,16]研究了多项式自函子如何产生模态逻辑.他们针对这些自函子的递归构造形式,定义其上的关系和谓词以及互模拟和不变式,以此产生模态.其研究成果可实际用于面向对象类规范语言,并已经用于验证 Java 类库的正确性及 Java 语言语义研究的许多其他方面[11~13];

(3) D. Pattinson[26,40]等人试图提出一种既方便应用又可用于一般函子的共代数逻辑.D. Pattinson 研究了如何通过自然变换从函子得到模态性,而 A. Kurz 研究了共代数中的模态逻辑与代数中的等式逻辑之间的对应关系[24].他们都试图以其他范畴(包括 Scott Topology, Accesible Category 等)为背景,研究适用于共代数的逻辑及其性质.

共代数和模态逻辑之间的联系正如代数与等式逻辑之间的联系.在泛代数理论中,一类代数是由等式来定义的.例如,群是由运算满足结合律、有单位元、每个元素有逆元等性质所定义.在泛代数理论中,Birkhoff 的著名定理建立了一类代数与等式逻辑之间的关系,即 Birkhoff 代数簇(varieties)定理:一类代数对于积、子代数和同态像封闭当且仅当它可以由一组等式所定义.该定理给出了代数上的逻辑性质与代数上通过同态射所进行的代数构造之间的联系,使用代数同态的方法给出了代数逻辑性质的描述,在泛代数理论中有着重要的地位,关于 Birkhoff 代数簇定理的具体内容,参见文献[41].

由于 Birkhoff 代数簇定理在泛代数理论中的重要性,因此不少学者研究该定理在共代数理论中的对应定理,即 Birkhoff 共代数簇(covarieties)定理(准确地说是共 Birkhoff 共代数簇定理).上述 3 种途径都是研究如何从函子产生模态语言,以描述共代数逻辑的性质,而 Birkhoff 共代数簇定理则是要研究的则是一组模态公式能定义什么样的一类共代数.对于从函子产生模态语言的方法,A. Kurz 指出,函子 F 产生的模态语言 $\mathcal{L}_F$ 都必须满足如下条件[42]:

(1) 任意 F-共代数$\langle A,\alpha\rangle$和 $\mathcal{L}_F$ 的公式存在语义推导关系$\models_{F,A}\subseteq A\times\mathcal{L}_F$;

(2) 该语义推导关系使得互模拟的元素在逻辑上等价,即对两个 F-共代数$\langle A,\alpha\rangle$和$\langle B,\beta\rangle$,若存在它们之间的互模拟关系 $R\subseteq A\times B$,使得$\langle x,y\rangle\in R$,则对$\forall\varphi\in\mathcal{L}_F, x\models_{F,A}\varphi\Leftrightarrow y\models_{F,B}\varphi$.

基于上述条件,A. Kurz 定义了一般情况下共代数的模态逻辑.进一步地,D. Pattinson 等人引入了共代数的模态语言的表现力等概念[26],证明了一定意义下的 Birkhoff 共代数簇定理,即一类共代数 $\mathcal{K}$ 对子共代数、共积共代数和同态像封闭当且仅当它可由一组模态公式所定义,而且这一组模态公式就是 $\mathcal{K}$ 满足的公式.

对于共代数簇,H. Gumm 等人从 F-态射的角度进行了研究,并引入了完备共代数簇[43](S. Awodey 等人更准确地称其为行为共代数簇 Behaviour Covarieties[44])的概念,研究了其有关性质.

S. Awodey, J. Hughes 等人进一步从共等式角度研究共代数簇定理[44],U. Wolter 还从共关系等角度讨论了共等式及共代数簇定理[45].A. Kurz[24],J. Adámek[46]等人研究了一般范畴作为基础范畴的共代数簇定理.A. Kurz 还研究了共代数的模态逻辑与代数的等式逻辑之间的对应关系[24].

4 共代数方法的应用研究

近几年来,共代数方法在计算机科学中得到了广泛应用,综合起来有以下几个方面:

4.1 共代数方法在自动机理论中的应用

J. Rutten 从共代数的角度考察了传统的自动机理论,包括互模拟关系、子共代数、商共代数等概念在自动机理论中的对应物,特别研究了共归纳原理(共归纳定义和共归纳证明方法)在自动机理论中的应用,还进一步研究了 Partial Automata[3].

另外,M. Pistore 的博士论文[47]及其他一些人研究了共代数方法在 History Dependent Automata(HD-自动机)理论中的应用.

4.2 共代数方法在并发模型及其语义研究中的应用

可以说,计算机科学领域的学者对共代数方法的关注起源于 P. Aczel 等人使用共代数方法给出了进程代数的语义[1].目前有许多学者对共代数方法在并发模型及其语义中的应用进行着研究,例如,J. Rutten 和 D. Turi 研究了并发的终止共代数模型的一般理论,特别是讨论了与相应的初始代数语义之间的关系[2].

J. Worrell 的博士论文[48]及发表的有关论文[49]从更广的角度(例如在 Sheaf categories 及 Topos 中)讨论了程序设计语言,特别是并发程序设计语言的终止语义.M. Lenisa 的博士论文[50]及发表的相关论文[51]是 P. Aczel 等人工作的继续,讨论了基于 Non-well-founded sets 或者说 hypersets 的共代数理论,并讨论了它们在进程代数、高阶并发程序设计语言、π-演算等语义研究中的应用.D. Turi 的博士论文[52]以共代数理论为工具研究了并发模型的操作语义和指称语义以及它们之间的关系.

4.3 共代数方法在面向对象语言及其语义研究中的应用

这一应用主要是 B. Jacobs 及其学生所进行的 LOOP(logic of object oriented programming)[13]项目.他们设计了基于共代数面向对象类规范描述语言 CCSL[31](coalgebraic class specification language),并设计和开发了原型系统,将使用 CCSL 语言描述的类规范转换为定理证明辅助系统 PVS 能识别的公式,还使用 PVS 验证类规范的性质,并对其进行求精和改进.

B. Jacobs 及其学生在这一领域发表了许多文章[8~17],如 B. Jacobs 首次指出了类、对象与共代数之间的关系[10];B. Jacobs 首次探讨了继承与共自由共代数之间的关系[9]等.

他们还讨论了使用共代数描述一系列比较大的应用系统(或类)实例,如内存管理[53]、序列结构[34]、资源管理的 Peterson 互斥算法[54]等,进一步研究了如何结合 CCSL 和 JML(Java modeling language)用于形式化描述 Java 语言的类,以将共代数理论用于 Java 语言的类库正确性验证及其他语义问题的研究[35,36].

4.4 共代数方法在代数规范及其语义研究中的应用

自从 J. Goguen 等人将泛代数理论用于研究数据类型的代数规范及其语义以来,除了描述数据类型的可构造性以外,数据类型的可观察性也一直是代数规范中研究的关键问题之一,为此人们也研究了代数规范的终结语义,但成果不多.近年来,J. Goguen 等人将代数规范发展到 Hidden Algebra 和 Hidden Logic,并将共代数中的概念应用到数据类型的规范描述,J. Goguen 的博士生 G. Roşu 的博士论文[29]详细讨论了这方面的研究成果.

由于共代数理论和代数理论的对应性,且从某种意义上说,代数理论适合描述数据类型的可计算性(构造性质),而共代数理论适合描述数据类型的可观察性质或行为,所以不少学者试图将代数规范和共代数规范结合起来,例如 C. Cîrstea 的博士论文[32]探讨了这一问题,A. Kurz 在其博士论文[24]中也讨论了如何结合代数及共代数方法描述面向对象的类规范.

目前,研究共代数方法的学者越来越多,也开拓了越来越多的研究方向.在共代数方法的应用方面,除了上述研究方向以外,新近引起人们注意的还有共代数方法在软件体系结构研究中的应用.比如,J. Rutten 在文献[55]中使用共代数方法描述软件体系结构中的连接件,描述一些连接件类型及连接件的复合;L.S. Barbosa 在文献[56]中使用共代数方法描述软件构件之间的比较、复合,并讨论了一些软件构件的性质等.

最近也有学者将共代数方法用于一些软件开发方法的研究,例如澳门联合国大学软件研究所(UNU/IIST)的 B. Aichernig 等人将共代数方法用于 RAISE 方法[57].H. Tews 的博士论文[58]探讨了如何将 UML 翻译成 CCSL,他认为使用共代数理论研究 UML 的语义或为 UML 提供新的描述方法也是一个值得注意的研究方向.

5 总结与展望

共代数作为代数的对偶概念,许多学者在研究共代数理论时都借鉴了泛代数理论的研究结果,许多共代数理论的结果也都与代数理论中的结果相对应.表 1 总结了迄今为止我们所见到的代数理论和共代数理论中的概念之间的对偶关系.该对偶关系是指相对应的概念在代数理论和共代数理论中的性质和地位对应,例如,共代数的共积与代数的积具有相似的性质等.

Table 1 The dual concepts between coalgebra and algebra
表 1 共代数与代数之间的对偶概念

Coalgebra	Algebra
Coproduct	Product
Subcoalgebra	Quotient algebra
Quotient coalgebra	Subalgebra
Bisimulation	?
Cocongruence	Congruence
Behavioural equivalence	Subalgebra
Largest behavioural equivalence	Smallest subalgebra
Observable behavior	Constructive element
Mono	Epi
Covariety	Variety
Final/Terminal coalgebra	Initial algebra
Cofree coalgebra	Free algebra
Coinduction	Induction
Modal logic	Equational logic
Right adjoint	Left adjoint

由于共代数理论还在不断发展,将来也许还会有越来越多的对偶关系被发现.展望共代数理论的进一步研究内容,我们认为至少有如下几方面:

5.1 共代数方法的范畴论基础研究

这一方向的主要研究内容是继续完善集合范畴上的共代数理论,并尽可能地将结果推广到一般范畴上去,特别是推广到 Scott 域范畴、Topos 等.其中研究的问题主要包括:

(a) 如何构造共代数极限的问题.探讨共代数的各种极限的存在条件和构造方法,特别是终结共代数的存在条件和构造方法.

(b) 子共代数结构的问题.探讨如何将基于集合范畴定义的子共代数等概念推广到一般范畴,以及在一般范畴中子共代数构成完备格的条件,是否构成代数格等.

(c) 共代数之间互模拟关系的结构.探讨如何将基于集合范畴定义的互模拟关系等概念推广到一般范畴,研究最大互模拟关系的存在条件,研究所有互模拟关系构成完备格的条件,研究描述直观意义上行为等价的其他概念等.

(d) 共归纳原理的数学基础.探讨共代数满足共归纳原理的条件.B. Jacobs, C. Hermida, A. Kurz 等人以 Fibration 理论为工具研究了共归纳原理和归纳原理统一的数学基础[14,59~61],但还有许多问题值得进一步探讨.

(e) 共代数理论的公理化.在这方面,A. Kurz 已做了初步的研究[24],但我们认为首先要探讨基于集合范畴的共代数理论的公理化,研究清楚共代数中哪些概念和结果是依赖集合的性质,然后进一步研究一般范畴上共代数理论的公理化.

5.2 共代数理论的逻辑基础研究

这一方向的主要研究内容是进一步研究共代数与模态逻辑之间的关系,特别是在一般范畴的基础上研究共代数和模态逻辑之间的关系,研究如何在共代数规范中使用模态逻辑描述共代数的性质,如何在共代数理论中使用模态逻辑的研究结果,如何将共代数理论应用到模态逻辑的研究中,研究共代数上的模态逻辑和代数上

的等式逻辑之间的对偶关系,研究模态逻辑公式所能定义的共代数类,研究各种 Birkhoff 共代数簇定理之间的关系,研究共代数上的其他逻辑等.

5.3 共代数理论的应用研究

前面总结了共代数在自动机理论、并发语义、面向对象程序语义、代数规范与共代数规范等方面的应用研究,但这些方面都还有许多内容值得进一步探讨.另外,将共代数理论用于软件体系结构和软件模式的形式基础研究,乃至用于对 UML 方法及其他软件开发方法的形式基础研究都值得进一步探讨.另外,我们注意到,共代数理论已用于描述 CSP,π-演算等并发模型的语义,那么,共代数理论是否也可作为研究 Petri 网的数学工具呢?这也是一个值得探讨的问题.

总之,共代数理论不仅可以使用数学工具(如范畴理论等)研究其数学基础,而且也可以进行广泛的应用研究.特别地,与代数理论类似,共代数方法不仅可用于程序语言的设计(例如在程序语言中提供共代数定义数据类型的手段)、程序设计语言语义的研究,也可以用于大型软件的规范描述及软件开发方法的形式基础研究.

References:

[1] Aczel P. Non-Well-Founded Sets. Standford: CSLI Press, 1988.

[2] Rutten J, Turi D. Initial algebra and final coalgebra semantics for concurrency. Technical Report, CS-9409, Amsterdam: Centrum voor Wiskunde en Informatica (CWI), 1994.

[3] Rutten J. Automata and coinduction: An exercise in coalgebra. In: Sangiorigi D, de Simone R, eds. Proceedings of the CONCUR'98. LNCS 1466, Springer-Verlag, 1998. 194~218. http://homepages.cwi.nl/~janr/papers/.

[4] Rutten J. Coalgebra, concurrency, and control. Technical Report, SEN-R9921, Amsterdam: Centrum voor Wiskunde en Informatica (CWI), 1999.

[5] Rutten J. Universal coalgebra: A theory of systems. Theoretical Computer Science, 2000,249(1):3~80.

[6] Turi D, Rutten J. On the foundations of final coalgebra semantics: non-well-founded sets, partial orders, metric spaces. Mathematical Structures in Computer Science, 1998,8(5):481~540.

[7] Reichel H. An approach to object semantics based on terminal co-algebras. Mathematical Structures in Computer Science, 1995, 5(2):129~152.

[8] Jacobs B. Mongruences and cofree coalgebras. In: Alagar VS, Nivat M, eds. Algebraic Methods and Software Technology. LNCS 936, Springer-Verlag, 1995. 245~260. http://www.cs.kun.nl/~bart/PAPERS/index.html.

[9] Jacobs B. Inheritance and cofree constructions. In: Cointe P, eds. European Conference on Object-Oriented Programming. LNCS 1098, Springer-Verlag, 1996. 210~231.

[10] Jacobs B. Objects and classes, co-algebraically. In: Freitag B, Jones CB, Lengauer C, Schek H.-J., eds. Object-Orientation with Parallelism and Persistence. Kluwer Academic Publishers, 1996. 83~103.

[11] Jacobs B. Invariants, bisimulations and the correctness of coalgebraic refinements. In: Johnson M, eds. Algebraic Methodology and Software Technology. LNCS 1349, Springer-Verlag, 1997. 276~291.

[12] Jacobs B, van den Berg J, Huisman M, van Berkum M, Hensel U, Tews H. Reasoning about classes in Java. SIGPLAN Notices, 1998,33(10):329~340.

[13] Hensel U, Huisman M, Jacobs B, Tews H. Reasoning about classes in object-oriented languages: Logical models and tools. In: Hankin CH, ed. European Symposium on Programming. LNCS 1281, Springer-Verlag, 1998. 105~121.

[14] Hermida C, Jacobs B. Structural induction and coinduction in a fibrational setting. Information and Computation, 1998,145(2): 107~152.

[15] Jacobs B. The temporal logic of coalgebras via galois algebras. Technical Report, CSI-R9906, Nijmegen: Computing Science Institute, 1999.

[16] Jacobs B. Towards a duality result in coalgebraic modal logic. In: Reichel H, ed. Proceedings of the CMCS 2000. Electronic Notes in Theoretical Computer Science Vol.33, 2000.

[17] Jacobs B. Many-Sorted coalgebraic modal logic: A model-theoretic study. Theoretical Informatics and Applications, 2001,35(1): 31~59.

[18] Adámek J. On final coalgebras of continuous functors. Theoretical Computer Science, 2003,294(5):3~29.

[19] Adámek J, Porst H.-E. From varieties of algebras to covarieties of coalgebras. In: Corradini A, Lenisa M, Montanari U, eds. Proceedings of the CMCS 2001. Electronic Notes in Theoretical Computer Science Vol.44, 2001,

[20] Gumm HP. Elements of the general theory of coalgebras. Lecture Notes, Rand Africans University, 1999. http://www.mathematik.uni-marburg.de/~gumm/Papers/publ.html.

[21] Gumm HP. Functors for coalgebras. Algebra Universalis, 2000,45(2-3):135~147.

[22] Gumm HP, Schröder T. Coalgebraic structure from weak limit preserving functors. In: Reichel H, ed. Proceedings of the CMCS 2000. Electronic Notes in Theoretical Computer Science Vol.33, 2000.

[23] Kurz A. Coalgebras and Modal Logic. Lecture Notes for ESSLLI 2001, 2001. http://homepages.cwi.nl/~kurz/works.html.

[24] Kurz A. Logics for coalgebras and applications to computer science [Ph.D. Thesis]. München: Institut für Informatik Ludwig-Maximilians-Universität München (LMU), 2000.

[25] Moss L. Coalgebraic logic. Annals of Pure and Applied Logic, 1999,96(1-3):277~317.

[26] Pattinson D. Expressivity results in the modal logic of coalgebras [Ph.D. Thesis]. München: Institut für Informatik Ludwig-Maximilians-Universität München (LMU), 2001.

[27] Pattinson D. Coalgebraic modal logic: soundness, completeness and decidability. Technical Report, München: LMU, 2002.

[28] Rößiger M. Coalgebras, clone theory, and modal logic [Ph.D. Thesis]. Fakultät Mathematik und Naturwissenschaften, Technischen Universität Dresden, 2000.

[29] Roşu G. Hidden logic [Ph.D. Thesis]. San Diego: University of California, 2000.

[30] Rothe J. Modal logic for coalgebraic class specification [MS. Thesis]. Institut Theoretische Informatik, Fakultät Informatik, Technischen Universität Dresden, 2000.

[31] Rothe J, Tews H, Jacobs B. The coalgebraic class specification language CCSL. Journal of Universal Computer Science, 2001,7(1):175~193.

[32] Cîrstea C. Integrating observatioins and computations in the specification of state-based, dynamical systems [Ph.D. Thesis]. Oxford University, 2000.

[33] Jacobs B, Rutten J. A tutorial on (co) algebras and (co) induction. EATCS Bulletin, 1997,62:222~259.

[34] Hensel U, Jacobs B. Coalgebraic theories of sequences in PVS. Journal of Logic and Computation, 1999,9(4):463~500.

[35] Jacobs B, Poll E. A logic for the Java modeling language JML. In: Hussmann H, ed. Fundamental Approaches to Software Engineering. LNCS 2029, Springer-Verlag, 2001. 284~299.

[36] Jacobs B, Poll E. Coalgebras and monads in the semantics of Java. Theoretical Computer Science, 2003,291(3):329~349.

[37] Gumm HP, Schröder T. Products of coalgebras. Algebra Universalis, 2001,46(1-2):163~185.

[38] Kurz A. Specifying coalgebras with modal logic. Theoretical Computer Science, 2001,260(1-2):119~138.

[39] Rößiger M. Coalgebras and modal logic. In: Reichel H, ed. Proceedings of the CMCS 2000. Electronic Notes in Theoretical Computer Science Vol.33, 2000.

[40] Pattinson D. Semantical principles in the modal logic of coalgebra. In: Proceedings of the 18th International Symposium on Theoretical Aspects of Computer Science. LNCS 2010, Springer-Verlag, 2001.

[41] Wechler W. Universal algebra for computer scientists. In: EATCS Monographs on Theoretical Computer Science, 1992.

[42] Kurz A. A co-variety-theorem for modal logic. In: Zakharyaschev M, Segerberg K, de Rijke M, Wansing H, eds. Proceedings of Advances in Modal Logic 2, Vol 2. Stanford University: CSLI Press, 2000.

[43] Gumm HP, Schröder T. Covarieties and complete covarieties. Theoretical Computer Science, 2001,260(1-2):71~86.

[44] Awodey S, Hughes J. The coalgebraic dual of birkhoff's variety theorem. Technical Report, CMU-PHIL-109, Department of Philosophy, Carnegie Mellon University, 2000.

[45] Wolter U. On corelation, cokernel, coequations. In: Reichel H, ed. Proceedings of the CMCS 2000. Electronic Notes in Theoretical Computer Science Vol.33, 2000.

[46] Adámek J, Porst H-E. On varieties and covarieties in a category. Mathematical Structure in Computer Science, 2003,13(2):201~232.

[47] Pistore M. History dependent automata [Ph.D. Thesis]. Universita di Pisa, 1999.

[48] Worrell J. On coalgebras and final semantics [Ph.D. Thesis]. Computing Laboratory, Keble College, Oxford University, 2000.

[49] Worrell J. Toposes of coalgebras and hidden algebras. In: Jacobs B, Moss L, Reichel H, Rutten J, eds. Proceedings of the CMCS'98. Electronic Notes in Theoretical Computer Science Vol.11, 1998.

[50] Lenisa M. Themes in final semantics [Ph.D. Thesis]. Universita Degli Studi di Pisa Dipartimento Di Informatica, 1998.

[51] Lenisa M. From set-theoretic coinduction to coalgebraic coinduction: Some results, some problems. In: Jacobs B, Rutten J, eds. Proceedings of the CMCS'99. Electronic Notes in Theoretical Computer Science Vol.19, 1999.

[52] Turi D. Functorial operational semantics and its denotational dual [Ph.D. Thesis]. Amsterdam: Free University, 1996.

[53] Tews H. Case study in coalgebraic specification: Memory management in the Fiasco microkernel. Technical Report, Institut Theoretische Informatik, Fakultät Informatik, Technischen Universität Dresden, 2000.

[54] Jacobs B. Exercises in coalgebraic specification. In: Backhouse R, Crole R, Gibbons J, eds. Algebraic and Coalgebraic Methods in the Mathematics of Program Construction. LNCS 2297, Springer-Verlag, 2002. 237~280.

[55] Rutten J. A coinductive calculus of component connectors. Technical Report, SEN-R0216, Amsterdam: CWI, 2002.

[56] Barbosa LS. Component as process: An exercise in coalgebra specification. In: Smith SF, Talcott CL, eds. Formal Methods for Open Object-Based Distributed Systems (FMOODS 2000). Kluwer Academic Publishers, 2000. 397~417.

[57] Meng S, Aichernig B. Component based coalgebraic specification and verification in RSL. Technical Report, No.267, United Nations University International Institute for Software Technology, 2002.

[58] Tews H. Coalgebra method for object-oriented specification [Ph.D. Thesis]. Institut Theoretische Informatik, Fakultät Informatik, Technischen Universität Dresden, 2002.

[59] Jacobs B. Comprehension for coalgebras. In: Moss L, ed. Proceedings of the CMCS 2002. Electronic Notes in Theoretical Computer Science Vol.65, 2002.

[60] Kurz A, Pattinson D. Notes on coalgebra, cofibration and concurrency. In: Reichel H, ed. Proceedings of the CMCS 2000. Electronic Notes in Theoretical Computer Science Vol.33, 2000.

[61] Hughes J, Jacobs B. Factorization systems and fibrations: Toward a fibred Birkhoff variety theorem. In: Blute R, Selinger P, eds. Proceedings of Category Theory and Computer Science 2002 (CTCS 2002). Electronic Notes in Theoretical Computer Science Vol.69, 2003.