计算机科学中的范畴数据类型的研究综述

苏锦钿
(华南理工大学计算机科学与工程学院 广州 510640)

摘 要 范畴数据类型是指以范畴论为数学理论基础研究数据类型的描述、计算、语义和应用。早期的范畴数据类型研究以归纳数据类型为主,采用代数从归纳的角度研究有限数据类型的构造语义和递归性质。近年来,归纳数据类型的对偶概念——共归纳数据类型逐渐引起计算机科学工作者的关注与研究,他们采用共代数从观察的角度研究无限数据类型的行为语义和共递归性质。利用范畴论可以为数据类型研究提供统一的数学理论基础,并将代数和共代数中的各种重要研究成果有机地融合在一起,如语法构造与动态行为、递归与共递归、同余与互模拟等。目前,范畴数据类型已经在程序语言、计算描述、理论证明器和并行计算等领域得到广泛的应用。对范畴数据类型的基本概念、数学理论基础、逻辑基础及应用等方面的最新研究成果进行介绍,以引起国内外相关研究领域的学者对计算机科学中的范畴数据类型理论的关注。

关键词 数据类型,范畴论,代数,共代数,逻辑演算,程序语言
中图法分类号 TP301.2 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.10.002

Survey on Categorical Data Type in Computer Science

SU Jin-dian
(College of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China)

Abstract Categorical data type refers to using category theory as the mathematical foundations for the researches on the description, computation, semantic and application of data types. The earlier work focuses on inductive data type and uses algebras to describe the construction semantics and recursive properties of finite data type from the inductive perspective. In recent years, coinductive data type, the dual concept of inductive data type, has attracted the attentions of many computer scientists, which use coalgebras to describe the behavioral semantics and corecursive properties of infinite data type from the observational perspective. Category theory can offer a unified mathematical theory foundation for various data types to discuss many important relations and properties, and integrate various important achievements of algebras and coalgebras, i.e. syntactic constructions and dynamic behaviors, recursion and corecursion, congruence and bisimulation and so on. Currently, categorical data type has been widely applied in many fields, such as programming languages, descriptions of calculation, theorem provers and parallel computations, etc. The latest research results of categorical data type in its basic concepts, mathematical foundations, logical foundations and their applications were introducted to attract the attention of the relative researchers.

Keywords Data type, Category theory, Algebra, Coalgebra, Logical calculus, Programming language

1 引言

作为形式语义及类型理论领域的一个新兴研究方向,范畴数据类型(Categorical Data Types,CDT)是指以范畴论为数学理论基础来研究数据类型的抽象描述、计算语义、构造或行为特征、性质及其在计算机科学中的应用等。早期对CDT的研究主要针对递归类型中的归纳数据类型,以从20世纪70年代开始Goguen、ADJ小组和Guttag等对抽象数据类型的代数语义的研究工作为代表,其思路是利用代数从归纳的角度刻画有限数据类型的构造语义和递归性质,并结合等式逻辑研究其代数规范和代数语义。因此,归纳数据类型通常也被称为代数数据类型,如自然数、有限分支树、链表和队列等都是典型的归纳数据类型。从80年代末开始,随着共代数方法(coalgebraic methods)在计算机科学中的广泛应用,一些学者结合非良基集理论提出了递归类型中的共归纳数据类型(coinductive data types)[1,2],其思路是利用共代数从共归纳的角度刻画无限数据类型的行为语义和共递归性质,并结合各种共代数逻辑研究数据类型的共代数规范和共代数语义。因此,共归纳数据类型也常被称为共代数数据类型或共数据类型(codata types)[3],如惰性数据类型中的无限分支树、无限链表和流等。在范畴论视角下,数据类型及其上的构造或观察操作都可以利用对象、射、函子和自然转换等概念给出统

到稿日期:2015-08-01 返修日期:2015-12-01 本文受广东省自然科学基金(2015A030310318),广东省医学科学技术研究基金项目(A2015065),国家自然科学基金资助项目(61103038)资助。
苏锦钿(1980—),男,博士,副教授,主要研究方向为形式语义、范畴论和自然语言理解。

一的描述，并通过同态射从递归或共递归的角度对数据类型上的计算函数进行抽象定义。其一方面为数据类型提供一种清晰简洁的语义，使研究者可以在统一的数学理论基础上研究它们的性质；另一方面，其对程序的设计、构造、转换、优化和证明等也具有重要的意义。例如，利用初始代数和归纳原理（或终结共代数和共归纳原理）可以给出数据类型上各种递归（或共递归）计算模式的抽象定义及其计算律。另外，利用范畴论中的对偶和伴随关系可以将长期以来在计算机科学中被广泛研究和应用的各种数据类型理论有机地结合起来，特别是代数和共代数中的众多重要研究成果，如构造语义和行为语义、归纳（或初始语义）与共归纳（或终结语义）、递归与共递归、指称语义和操作语义、同余与互模拟、代数逻辑与共代数逻辑等，并研究不同范畴或函子上的数据类型间的关系及性质，或将特定范畴上的数据类型研究扩展到其他范畴上。可以说，CDT 所展现出来的强大生命力和在程序语言设计、计算描述、理论证明器、并行计算和泛化编程等众多计算机科学领域中的广阔应用前景，特别是它所引入的程序设计方法、理解数据类型的新思路和对数据类型及其描述计算的高度抽象性等特点，已经引起了研究人员的极大兴趣。

1987 年，Hagino 最早在文献[4]中指出范畴论比集合论更能自然地描述和理解数学对象，并提出了初始（或归纳）数据类型和终结数（或共归纳）据类型，统称为范畴数据类型。Hagino 的研究思路是将归纳和共归纳数据类型分别看成是分配范畴上的共变函子（covariant）的初始代数和终结共代数。随后，Hagino[5]利用 CDT 中的终结数据类型对函数式程序语言 ML 进行扩展，并给出了相应的范畴论及域论的解释。Wraith[6]则在 Hagino 的基础上将分配范畴简化为双笛卡尔封闭范畴，并结合双函子给出参数化 CDT 的代数和共代数描述。这些工作初步奠定了 CDT 中归纳和共归纳数据类型的代数和共代数数学理论基础。

90 年代初，Malcolm[7]将 Hagino 的研究工作引入到基于代数的程序演算研究中。Meijer[8]和 Geuver[9]等分别从函数式程序语言和多态类型化 lambda 演算的角度对归纳和共归纳数据类型上一些典型计算进行分析。Greiner[10]探讨了共归纳数据类型与归纳数据类型间的对偶性，并给出了共归纳数据类型上的各种共递归计算及在程序中的应用。Cockett 等[11]在 Hagino 的研究基础上利用强函子将 CDT 扩展为强 CDT（包括强归纳和强共归纳数据类型），并将其应用于函数式编程语言 Charity。这促使 CDT 开始应用于程序语言的设计、转换和优化。Skillicorn 等[12,13]则利用 CDT 上操作的并行性和与机器实现及体系结构无关的性质构造抽象的并行机模型，并将其应用于并行计算中。

从 20 世纪 90 年代中后期开始，越来越多的学者开始对 CDT 的数学理论基础、逻辑基础及应用进行深入的研究，如 Fokkinga[14]和 Erwig 等[15]对 CDT 的双代数结构的研究，Vesely 等[16]对基于 CDT 的参数化编程或泛化编程的研究，Parod 等[17,18]对强 CDT 及带参数的递归计算的研究，Uustalu[19]，Cunha[20]及 Dominguez 等[21]对各种递归和共递归及其计算律的研究，Uustalu[22]和 Vene[23]对 Mendler-风格归纳及共归纳数据类型的研究，Cirstea[24]和 Leivant[25]等对 CDT 中共代数逻辑的研究，Bird[26]，Hinze[27]，Blampied[28]和 Abel[29]等对嵌套数据类型的研究，Traytel[30]等对 CDT 在高阶逻辑和理论证明器中的应用研究。目前，这一领域的研究正在不断的发展，上面只列出其中一些具有代表性的工作。

我们发现，对于计算机科学中的 CDT，目前的研究主要集中在其数学理论基础、逻辑基础及应用等 3 方面。因此，本文就这几方面对这一领域的研究工作进行综述。本文第 2 节简要介绍 CDT 的定义，并结合部分典型的例子进行说明；第 3 节综述 CDT 在理论基础方面的研究；第 4 节综述 CDT 在逻辑基础方面的研究；第 5 节综述 CDT 在计算机科学中的应用；第 6 节总结全文，并展望 CDT 的发展方向。受篇幅所限，本文内容不可能包含这一领域的所有研究成果，只能概述其中主要的代表性研究工作和成果，希望能够促使研究程序语言设计、类型理论研究、逻辑学者等领域的读者开始关注并研究这一理论。

2 范畴数据类型的定义

为了便于后面的介绍，下面先给出代数及共代数的范畴论定义。范畴论的相关介绍可见文献[31]。假定下面的范畴 $\mathbb{C}$ 包含积、共积及终结对象。

定义 1 给定范畴 $\mathbb{C}$ 和自函子 $F:\mathbb{C}\to\mathbb{C}$，函子 F 上的代数定义为二元组 $(X,\alpha_X:FX\to X)$，其中 X 是 $\mathbb{C}$ 中的对象，称为该 F-代数的载体；α_X 是 $\mathbb{C}$ 中的射，称为该 F-代数的基调。任意两个 F-代数 (X,α_X) 和 (Y,α_Y) 间的同态射 $f:(X,\alpha_X)\to(Y,\alpha_Y)$ 是 $\mathbb{C}$ 中的射 $f:X\to Y$，且使得图 1 中的图表满足交换条件。称 $(\mu F,\mathrm{in}_F:F\mu F\to\mu F)$ 为初始 F-代数，当且仅当对任意一个 F-代数 (X,α_X)，存在唯一射 $\mathrm{fold}_F(\alpha_X):\mu F\to X$，使得图 2 中的图表满足交换条件。

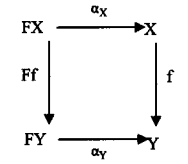

图 1 代数同态射

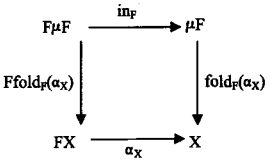

图 2 初始代数及唯一射

$\mathrm{fold}_F(\alpha_X)$ 是由基调 α_X 所确定的迭代射，也称为 fold 或 catamorphism[32]。在计算机科学中，常用代数理论来研究有限数据类型的构造，其中代数函子给出了数据类型上各种构造操作的抽象描述。通过选择合适的范畴和函子，许多抽象数据类型恰好就是某个代数函子的初始代数，且为递归类型等式的最小固定点。

例 1 函子 $\mathrm{ListF}(X)=1+A\times X$ 的初始 ListF-代数 $(A^*,[\mathrm{nil},\mathrm{cons}])$ 是一个包含类型为 A 的元素的数组 A^*，其中初始化操作 $\mathrm{nil}:1\to A^*$ 和插入操作 $\mathrm{cons}:A\times A^*\to A^*$ 为 A^* 的构造子。任意一个数组 A^*，如 $[\langle\rangle,\mathrm{cons}(a_1,\langle\rangle),\mathrm{cons}(a_2,\mathrm{cons}(a_1,\langle\rangle)),\cdots,\mathrm{cons}(a_n\cdots\mathrm{cons}(a_1,\langle\rangle))]$，可在有限步骤内迭代地应用 nil 和 cons 构造而成。由初始代数的初始性可得到：对任意一个 ListF-代数 $(X,[f,g]:1+A\times X\to X)$，都存在唯一射 $(|[f,g]|)_{ListF}:A^*\to X$，使得满足：$fold_{ListF}([f,$

$g]) \circ \text{nil} = f$ 和 $fold_{ListF}([f,g]) \circ \text{cons} = g \circ Id_A \times fold_{ListF}([f,g])$。

要定义数组 A^* 上的求长度函数 length: $A^* \to$ nat，可先定义 nat 上的两个操作 zero: $0 \to$ nat 和 succ: nat $\to$ nat，使得满足等式：length(nil) = 0 和 length(cons(a, L)) = succ(length(L))。显然，length 是一个 ListF-代数同态射 length: (A^*, [nil, cons]) $\to$ (nat, [zero, succ $\circ \pi_2$])，其中 $\pi_2 : A \times \text{nat} \to \text{nat}$ 为笛卡尔积上的一个投影射，succ $\circ \pi_2 : A \times \text{nat} \to \text{nat}$ 使得 nat 具有与 A^* 相同的操作类型，即为同一函子下的不同基调。

类似地，其他的抽象数据类型，如树、堆栈、集合和队列等，也都可利用代数进行描述。由初始代数的初始性可得到代数中非常重要的归纳原理（包括归纳定义原则和归纳证明原则），并且可以给出归纳数据类型上许多固定模式的递归函数的抽象定义，如迭代、原始递归（也称 paramorphism）、Course-of-Value 迭代（也称 histomorphism）和互迭代等。这些递归函数及其计算律在程序的定义、计算、转换、优化及证明等研究中具有非常重要的作用。

代数方法适合从归纳的角度研究有限数据类型的构造语义，在刻画无限数据结构及其行为语义方面则存在不足。像流、无限分支树或数组等这一类的无限数据类型可通过代数的范畴对偶概念——共代数进行描述，其中数据类型上满足一定性质的观察操作可看成是某个共代数函子的基调，而共代数函子的终结共代数则给出共归纳数据类型的定义。

定义 2 给定范畴$\mathbb{C}$和自函子 $B: \mathbb{C} \to \mathbb{C}$，函子 B 上的共代数定义为二元组 $(X, \beta_X : X \to BX)$，其中 $X \in \mathbb{C}$ 称为该 B-共代数的载体或状态空间，射 β_X 称为该 B-共代数的结构射。任意两个 B-共代数 (X, β_X) 和 (Y, β_Y) 间的同态射 $f: (X, \beta_X) \to (Y, \beta_Y)$ 是$\mathbb{C}$中的射 $f: X \to Y$，且使得图 3 中的图表满足交换条件。称 $(\nu B, \text{out}_B : \nu B \to B\nu B)$ 为终结 B-共代数，当且仅当对任意一个 B-共代数 (X, β_X)，存在唯一射 $\text{unfold}_B(\beta_X): X \to \nu B$，且使得图 4 中的图表满足交换条件。

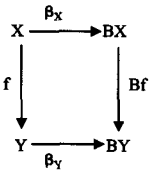

图 3 共代数同态射

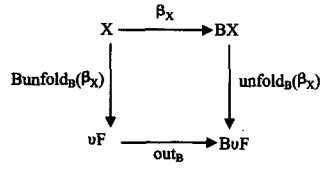

图 4 终结共代数及唯一射

$\text{unfold}_B(\beta_X)$ 是由基调 β_X 所确定的共迭代射或共归纳扩展射，也称为 unfold[32] 或 anamorphism[8]。在计算机科学中，常用共代数理论来研究无限数据类型的行为语义，其中共代数函子给出数据类型上各种观察操作的抽象描述。通过选择合适的范畴和函子，许多无限数据类型恰好就是某个共代数函子下的终结共代数，且为递归类型等式的最大固定点。

例 2 对数组 A^* 的观察包括：谓词操作 isnil: $A^* \to 1+1$ 用于判断数组是否为空，以及观察操作 head: $A^* \to A$ 和 tail: $A^* \to A^*$ 用于分别给出数组的当前元素及剩余部分。这些观察操作可看成是共代数函子 $\text{ListB}(X) = 1 + A \times X$ 的一个基调，并表示为 ListB-共代数：(A^*, ⟨isnil, head, tail⟩: $A^* \to 1 + A \times A^*$)。由终结共代数的终结性有：对任意一个 ListB-共代数 $(X, \langle f, g, h\rangle : X \to 1 + A \times X)$，都存在唯一射 $\text{unfold}_{\text{ListB}}(\langle f, g, h\rangle): X \to A^*$，使得满足：$\text{isnil} \circ \text{unfold}_{\text{ListB}}(\langle f, g, h\rangle) = f$ 以及 $\text{head} \circ \text{unfold}_{\text{ListB}}(\langle f, g, h\rangle) = g$ 和 $\text{tail} \circ \text{unfold}_{\text{ListB}}(\langle f, g, h\rangle) = g \circ \text{unfold}_{\text{ListB}}(\langle f, g, h\rangle)$。

由终结共代数的终结性可以得到共代数中非常重要的共归纳原理（包括共归纳定义原则和共归纳证明原则），并且可给出共归纳数据类型上许多固定模式的共递归函数的抽象定义，如共迭代、原始共递归（或称 apomorphism）和 Course-of-Value 共迭代（或称 futumorphism）[9] 等。这些共递归函数及其计算律在基于行为关系的函数定义和程序推理、转换和优化过程中具有重要的作用[3]。利用共归纳数据类型还可将共代数中的互模拟、终结语义和共归纳原理等理论引入到数据类型及程序语言设计研究中，增强程序语言对动态行为的描述和验证能力，并与归纳数据类型研究形成互补。事实上，归纳数据类型与共归纳数据类型中的许多概念是范畴互对偶。

近年的研究发现，许多数据类型除了具有可递归定义的语法构造或静态结构特征外，在计算过程中还会展现出可共递归定义的观察效果或动态行为特征。例如，在程序计算中，一方面需要构造树或堆栈，另一方面需要观察树的各个分支上的标签或堆栈中的栈顶元素。即这一类的数据类型同时具有归纳和共归纳数据类型的典型特征。因此，单纯利用代数或共代数难以全面地刻画它们，也无法进一步研究它们间的关系及性质。为解决该问题，一些学者尝试对代数进行扩展或者将代数与共代数结合起来。一些主要的工作包括 Hagino[4] 对基于 dialgebras 的 CDT 研究，Goguen[33] 等对基于 hidden algebra 的数据类型行为规范的研究及 Fokinga、Erwig 和 Nogueira[14,34,35] 等对基于双代数的 CDT 研究。

所谓的⟨F, B⟩-双代数是指同一载体集上的代数和共代数对 $(X, \alpha_X : FX \to X, \beta_X : X \to BX)$。在双代数研究中通常只考虑满足某些合适条件的双代数结构，如存在某种分配律 λ（如 $\lambda: FB \Rightarrow BF$）的 λ-双代数。这种 λ-双代数通过分配律 λ 将双代数结构中的代数构造操作和共代数行为变迁有机地融合在一起。简单地说，一方面，代数中的构造操作是具有良行为的操作；另一方面，共代数上的行为变迁在这些代数构造操作下保持。分配律 λ 不仅可以给出构造操作与产生合适行为之间的关系，还可以通过函子化提升研究递归与共代数结构、共递归与代数结构之间的关系及性质，从而增强代数中的归纳原理或共代数中的共归纳原理。

CDT 中任意的归纳或共归纳数据类型都可以通过适当的扩展而构成一个双代数结构。根据 Lambek 定理和文献[36]中定理 9.1 可知初始代数和终结共代数中的基调均为同构射，因此利用它们的逆可构造对应的双代数结构。例如，代数函子 $FX = 1 + X$ 的初始代数 $(\mu F, \text{in}_F = [\text{zero}, \text{succ}])$ 给出了归纳数据类型 nat（即自然数）的定义，逆 $\text{in}_F^{-1} = \langle \text{iszero}, \text{pred}\rangle$ 中的谓词操作 iszero: nat $\to 1+1$ 用于判断自然数是否为 0，pred: nat $\to$ nat 给出了某个非零自然数的前驱。因此，(nat, $\text{in}_F = [\text{zero}, \text{succ}]$, $\text{in}_F^{-1} = \langle \text{iszero}, \text{pred}\rangle$) 是 nat 上的一个双代数结构。类似地，共代数函子 $BX = \text{int} \times X$ 的终结共代数给

出了共归纳数据类型中整数流 int^{∞} 的定义，基调 $\text{out}_B = \langle \text{value}, \text{next} \rangle$ 分别给出整数流的当前值及后继状态，逆 $\text{out}_B^{-1} = [\text{scons}]$ 中的操作 $\text{scons}: \text{int} \times \text{int}^{\infty} \to \text{int}^{\infty}$ 表示将一个整数作为前缀添加到流中。因此，$(\text{int}^{\infty}, \text{out}_B^{-1} = [\text{scons}], \text{out}_B = \langle \text{value}, \text{next} \rangle)$ 是整数流 int^{∞} 上的一个双代数结构。

3 范畴数据类型的数学理论基础研究

从 20 世纪 90 年代中后期开始，CDT 在理论基础研究方面取得了较大的进展。下面从计算机科学的角度出发将这些研究归纳为以下 3 个方面。

3.1 CDT 的数学描述研究

CDT 的基本研究思路是将数据类型看成是某个合适范畴中的对象，其上的构造（或观察）操作看成该范畴的某个代数（或共代数）共变自函子的基调。数据类型上的共变递归类型等式的最小解和最大解分别对应着归纳和共归纳数据类型。因此，这方面主要研究数据类型在范畴论中的数学解释，特别是各种 CDT 与基范畴及函子间的关系、归纳数据类型的代数描述和其初始代数语义、共归纳数据类型的共代数描述及其终结共代数语义等。

许多学者通过对基范畴或函子添加各种约束条件以保证递归类型等式存在最小和最大解，如 Hagino[4]、Cockett[11] 和 Pardo[17,18] 等对基于分配范畴及强共变函子的 CDT 研究；Wriaith[6] 对基于双笛卡尔封闭范畴和双函子的 CDT 研究；Erwig[37] 对基于分配范畴及共变函子的 CDT 研究；Fokkinga 和 Meijer[38] 等对完全偏序范畴 CPO 和多项式函子的 CDT 研究。一些学者如 Cockett[39] 和 Vesely[16] 等还进一步从 2-范畴、2-函子和 2-伴随等更抽象的角度分析了 CDT 的性质和计算。例如，Charity 中的强数据类型实际上就是 Vesely[16] 所提出的 2-范畴数据类型的实例。

在 Hagino 等人的研究基础上，Uustalu 指出传统基于初始代数（或终结共代数）的归纳（或共归纳）数据类型描述难以扩展到非共变函子上，并将 Mendler 对类型理论中归纳数据类型的研究扩展到范畴论中，提出基于 Mendler-风格的归纳数据类型[22]。Bird 等[40] 将参数化归纳数据类型的递归声明中的固定参数扩展为可变参数，提出嵌套数据类型（nested datatypes）。Chuang[41] 利用代数给出代数依赖数据类型的范畴论解释。Capretta[42] 将共代数中的共递归概念引入到代数中，并提出共递归代数和广义共递归。而 Bonsangue[43] 则给出共归纳联合类型间语义子类型的范畴论解释。这些工作进一步丰富了 CDT 的研究，并促使范畴论应用于其他更复杂的数据类型建模。笔者在前期工作中也分析了程序语言中共归纳数据类型及 CDT 上子类型关系的范畴论解释[2,44]。

目前，这方面还有很多工作有待完善，特别是 CDT 与基范畴及函子之间的关系。需要将共完备范畴（如集合范畴、分配范畴或完全偏序范畴等）上的 CDT 研究扩展到其他范畴上，如研究 presheaf 范畴上的索引归纳和共归纳类型或 monoidal 范畴上 CDT 的并行性；研究其他各种复杂的 CDT（如互递归/互共递归数据类型、弱/强 CDT、非代数或共代数数据类型、嵌套数据类型或依赖数据类型）与函子类型（如共变函子、强函子、非共变函子或混合变函子等）之间的关系及性质。例如，如何将集合范畴上的多项式函子扩展到其他范畴上以描述互递归数据类型，或者研究格理论中非单调代数函子 F 的固定点与 μF 之间的关系，以及共变函子的递归类型等式的最小解（或最大解）为归纳（或共归纳）数据类型的条件。另外，分析和比较 CDT 与其他数据类型理论间的关系也很重要，特别是代数规范、域论或类型理论等，这有助于更好地理解 CDT 的特点，而且也便于融合其他数据类型理论的优点。例如，集合范畴上的 CDT 可看成是集合论上的类型理论研究的扩展，那么完全偏序范畴上的 CDT 是否与域论上的数据类型研究也存在类似的关系呢？

在 Mendler-风格的 CDT 研究方面，虽然 Uustalu 和 Vene 已初步给出了 Mendler-风格的归纳和共归纳数据类型的范畴论解释，但仍存在许多问题尚未很好地解决，如混合变函子上的初始 Mendler-风格代数和终结 Mendler-风格共代数存在的条件和性质、与传统的归纳或共归纳数据类型间的对应关系和转换条件等。在嵌套数据类型方面，需要进一步研究嵌套数据类型上各种复杂声明（如包含函数空间、变量绑定、高阶或多类）的范畴论解释以及嵌套数据类型等式存在最小解或最大解的条件和性质，并将目前对嵌套归纳数据类型的研究扩展到嵌套共归纳数据类型上。例如，嵌套归纳数据类型中包含了某种结构不变量，那么嵌套共归纳数据类型中是否也类似地包含了某种行为不变量呢？

3.2 CDT 的计算律研究

这方面是 CDT 的研究重点，主要是利用 CDT 的某些特殊性质（如初始代数或终结共代数的泛性质和唯一性）研究数据类型上的各种递归或共递归的计算律（如消去、反射和融合等）。一些典型的相关工作包括 Geuvers[9] 和 Dominguz 等[21] 对原始递归和广义 paramorphisms 的研究、Uustalu 和 Vene[19] 对原始递归/共递归和 Course-of-Value 迭代/共迭代的研究、Miranda-Perea[45] 对 Course-of-Value 递归的研究、Cunha 等[20] 对 hylomorphism 的研究、Parod[17,18] 等对带参数递归计算的研究、Uustalu[22] 和 Vene[23] 对 Mendler-风格归纳数据类型上各种递归及其计算律的研究、Jeannin[46] 对共归纳数据类型上递归函数的研究、Hinze 等[47] 对 hylomorphism 的研究和 Clouston 等[48] 对共归纳数据类型上受限递归的研究等。笔者在文献[1,49-51]中也分别研究了参数化共归纳数据类型上的共递归计算及带参数的 hylomorphisms。文献[8,19,23]对一些常见的递归或共递归及其计算律进行了较为全面和详细的介绍。这些研究主要针对一般归纳数据类型上的递归或共归纳数据类型的共递归，很少考虑其他各种更复杂的归纳或共归纳数据类型（如弱归纳或弱共归纳数据类型、满足等式或共等式定律的非归纳或非共归纳数据类型）上的计算，也没分析其他一些更复杂的计算模式（如 guarded 共递归、混合互递归或累积共递归等）。Gibbons 和 Hutton 等[32] 给出集合范畴中 CDT 上的函数构成 fold 或 unfold 的充分必要条件，但仍需将其进一步扩展到其他范畴及其他类型的 CDT 上。

一些学者将一般的递归或共递归研究扩展到嵌套数据类型上。Bird 在文献[40]中给出嵌套数据类型上的 fold 计算的一种高阶描述。随后，Bird 和 Paterson[26] 指出由嵌套数据类型的初始代数所确定的 fold 不足以描述嵌套数据类型上一些常见的结构化递归模式，并系统化地给出每一个嵌套数据类型上广义 fold 的构造方法及相应的计算律，还证明了这些广义 fold 与一般的 fold 一样具有相应的唯一性性质。在

Bird 的研究基础上，Hinze[27] 给出嵌套数据类型上的多型(polytypic)函数及约简函数，Blampied[28] 给出嵌套数据类型的结构化递归性质，而 Abel 等[52] 则利用系统 F^{ω} 给出任意有限秩的高阶嵌套数据类型上的迭代和共迭代及相应的 Mendler-风格描述。目前这方面还有很多问题有待解决，例如，嵌套归纳数据类型上是否存在与一般归纳数据类型相似的初始语义、各种递归计算模式和计算律？归纳数据类型与嵌套归纳数据类型上各种递归计算间的对应关系、条件、性质和计算效率的高低是如何的？如何将嵌套归纳数据类型的研究扩展到嵌套共归纳数据类型上？

部分学者结合一些抽象的数学结构(如 monads 或 comonads)给出各种递归或共递归的结构化描述，并统一地研究它们的抽象性质及其计算律，例如 Pardo[18] 对基于积 comonad 的强 CDT 上带参数(包括固定参数和累积参数)的递归研究、Uustalu 等[53] 对基于共自由递归 comonads 的递归研究、笔者[49,54] 对强共归纳数据类型上基于积 comonad 的广义共迭代及其计算律的研究。相对于一般的递归(或共递归)，基于 monads 或 comonads 的递归(或递归)一方面能够给出一种统一的结构化描述，并通过实例化得到各种具体的递归(或共递归)；另一方面，其还具有一些良好的性质，例如，由 comonads 所生成的递归共代数比归纳数据类型上的初始语义更适合于描述结构化递归关系，且更加抽象[55]。已有不少学者对这方面开展了相关研究，但仍需将上述研究扩展到一些更复杂的计算模式(如累积计算、循环定义、混合递归或共递归等)或 CDT(如 Mendler-风格、依赖和嵌套归纳或共归纳数据类型等)；另外，将 monads 对计算副作用和 comonads 对计算上下文的结构化描述有机地结合起来，用于提高程序计算的模块化和结构化也需要更多的研究工作。一些学者已在这方面做了一些前期工作，如 Pardo[56] 对 monadic 共递归及 monadic hylomorphisms 的研究。笔者在前期工作[49,54,57] 中也对这方面进行了一些探讨，主要包括强归纳数据类型上带固定参数且产生计算副作用的递归操作的 monadic 描述[57]、强共归纳数据类型上 unfold 和带固定参数的 unfold 操作的 comonadic 描述[54] 和广义共迭代[49]。

但这些研究工作仍未能充分地利用 monads 和 comonads 的优点，特别是 monads(或 Kleisli 范畴)和 comonads(或 coKleisli 范畴)以及 monads 代数和 comonads 共代数间的范畴对偶性及它们所满足的特殊性质。

除了上述工作，一些学者还对递归及共递归之间的关系进行了研究。例如，Gianantonio[58] 利用良基拓扑上的 sheaf 范畴的固定点理论拓给出了混合递归和共递归的定义，Hinze[59,60] 和 Cockett[39] 等从伴随的角度对递归及共递归之间进行分析。C. Fumex[61] 等利用范畴间的纤维性质研究了索引归纳和共归纳数据类型间的关系。

3.3 构造语义和行为语义研究

这方面主要是研究数据类型上的构造和观察操作的统一描述，以及递归与共递归、构造语义与行为语义、同余与互模拟等之间的关系和性质。部分典型的代表性工作包括：Hagino[4]，Glimming[62] 和 Goguen[33] 等从代数扩展的角度对 CDT 的研究，以及 Fokkinga[14]，Erwig[15,34,37] 和 Nogueira[35] 等从双代数的角度对 CDT 的研究。

T. Hagino 等[4] 最早利用 $\langle F,G\rangle$-dialgebras 结构统一地描述归纳和共归纳数据类型。随后，Glimming[62] 给出如何将某个固定参数化数据类型上的迭代或共迭代函数转化为该数据类型上的 di-递归和原始 di-递归。虽然 Dialgebras 可以给出代数和共代数的一种统一描述，但正如 Vene[23] 所指出：dialgebras 可以解决 CDT 上的混合变函子的问题，代价是无法明确地区分归纳及共归纳数据类型。Goguen[33] 则利用 hidden algebras 对数据类型上的行为规范进行研究。这些研究主要是通过对代数进行扩展并增加对共归纳原理的支持，从而可用于研究无限数据类型及其行为关系。

为了充分利用代数在描述数据类型的构造语义和共代数在描述数据类型的行为语义方面的优势，Fokkinga[14]，Erwig[15,34,37] 和 Nogueira[35] 等将双代数应用于 CDT 的描述中。Fokkinga 在利用范畴论研究数据类型上的计算律时指出：堆栈等数据类型中包含了构造操作和 destructor 操作，可以表示为双代数[14]。随后，Erwig[15,34,37]，Nogueira[35] 和笔者[63,64] 均给出了抽象数据类型在 CPO 范畴上的双代数结构。Erwig 将程序描述为一个抽象数据类型中的共代数结构到另一个抽象数据类型中的代数结构的映射，并提出一种 metamorphic 编程风格[15,37]，其本质是对 hylomorphisms 研究的扩展。Nogueira[35] 侧重于研究抽象数据类型的双代数结构在多态编程中的应用。而笔者[63,64] 则主要利用 λ-双代数探讨了构造操作与观察操作、迭代与共迭代间的关系及性质。除此之外，一些学者还利用 λ-双代数研究广义的递归和共递归。例如，Bartels[65] 利用 λ-双代数给出一种广义共归纳(称为 λ-共迭代)用于抽象地描述各种典型的共递归。Cancila[66] 在 Bartels 的研究基础上，给出基于双代数和 monads (T,η,μ) 的 T-共迭代定义，并将其作为各种共迭代(包括原始共递归、Course-of-Value 共迭代和 mutual 共迭代)的抽象描述。

近年来，双代数成为共代数领域的一个研究热点，在数据类型研究领域也具有广阔的应用前景和研究价值。双代数可以为数据类型上的构造语义和行为语义研究提供一个统一的数学理论基础，并通过分配律将代数和共代数有机地融合在一起，既增强了初始代数上的归纳原理，又增强了终结共代数上的共归纳原理。我们认为这将对今后的数据类型研究及程序语言设计产生深远的影响。目前，这方面的研究才刚刚起步，许多问题还有待解决和完善，例如数据类型上的构造和观察操作构成双代数结构的前提及性质(包括代数结构上的共代数构造、共代数结构上的代数构造，以及代数或共代数的函子化提升等)和函子间分配律的定义、性质、存在性条件或函子化提升等；另外，基于良基集的归纳原理与基于非良基集的共归纳原理、各种互模拟关系(如基于上下文或互相似的互模拟、弱互模拟、迹等价、错误等价)与代数同余之间的关系及性质等问题也很值得研究。而且，需要将现有的研究扩展到其他的 CDT 上。

4 范畴数据类型的逻辑基础研究

与代数规范中利用等式逻辑描述数据类型上代数基调的构造语义类似，数据类型上的共代数基调操作也必须满足一定的约束条件才能构成完整的行为语义描述。目前，利用共代数描述共归纳数据类型的动态行为已经得到广泛的认同，但采用什么样的逻辑来描述共代数的行为却还没有定论。因

此，这方面主要是研究适用于 CDT 的逻辑系统，特别是共归纳数据类型上的逻辑以及与代数中等式逻辑之间的关系。中山大学的周晓聪副教授已经在文献[67]中对共代数的逻辑研究进展进行了详细的介绍。这里只介绍与 CDT 相关的一些代表性研究工作。

概括起来，目前 CDT 中采用的共代数逻辑主要包括如下几方面。

(1)等式逻辑

一些学者直接利用代数中的等式逻辑来描述共归纳数据类型的行为语义。例如，Hagino[4]利用代数规范中的条件等式给出了 CDT 中的函子和自然转换必须满足的性质，Cockett[11]在研究 Charity 中强 CDT 的描述及推导时所采用的项逻辑实际上也是基于等式关系。类似地，Leivant 等[25]对原始共递归特性的研究也是利用等式逻辑。Hagino 在其研究中[4]指出：尽管范畴论中的许多概念都能够从等式关系的角度进行描述，但这不能完全体现范畴论及伴随的优点。而且，数据类型上的行为等价关系体现的是观察不可区分性，而不是严格的等式关系。

(2)共等式及共簇

一些学者利用共自由共代数给出等式逻辑及代数簇的对偶概念——共等式(coequation)和共代数共簇(covariety)，并将其应用于共归纳数据类型的全局行为描述。例如，Cirstea[24]利用共等式描述共归纳数据类型中同一状态上的各种不同观察间的等价性。总体来说，共等式并未像泛代数中的等式逻辑一样得到广泛的应用。

(3)模态逻辑

模态逻辑是共代数逻辑的研究重点。由于模态逻辑中可以包含各种模态词，因此模态逻辑比其他的逻辑系统更适合用于描述共代数的行为语义，特别是从局部的角度描述单个状态上的观察性质[68]。许多学者从 Stone 对偶、谓词提升和自然转换等不同的角度分别探讨了如何由共代数产生合适的模态逻辑。Schroder 和 Mossakowski 等人[69]将共代数及模态逻辑引入到代数规范语言 CASL 中，并研究共归纳数据类型上的模态操作符。

除上述研究外，还有一些学者采用了其他的逻辑。例如，Simon[70]从最小固定点理论及最小 Herbrand 模型的对偶角度出发，利用共归纳和最大 co-Herbrand 模型对传统基于归纳的逻辑编程进行扩展，提出了共归纳逻辑编程，并结合共归纳谓词和 Horn 子句对无限或循环数据类型进行描述和证明。Hasuo[71]则利用纤维性研究了终结共代数上的共归纳谓词及构造逻辑。

目前，这方面的研究才刚刚起步，如何由 CDT 上各种函子(如有限和无限 Kripke 函子、二元函子、非共变函子或混合变函子等)与产生合适的模态逻辑、各种逻辑构造语法和组合子在 CDT 中的解释以及模态逻辑与等式逻辑、共代数共簇与代数簇之间的关系等众多问题还有待解决。

该领域需要解决的另一个关键问题是如何将代数中的等式逻辑和各种共代数逻辑有机地融合在一起，给出适合 CDT 且包含构造语义和行为语义描述的逻辑系统。一些学者已经在这方面做了一些研究，如 Mossakowski 等[72]对 CoCASL 中基于等式逻辑的代数规范和基于模态逻辑的共代数规范之间关系的研究以及 Klin 等[73]对如何将共代数中的模态逻辑扩展到双代数中的研究。目前，许多基于代数的形式规范已经开始逐渐支持共代数和共归纳原理，但对等式逻辑和各种共代数逻辑之间的融合仍面临许多问题，特别是如何给出描述代数构造操作的模态词以及满足一定性质的各种逻辑分配律。

5 范畴数据类型的应用研究

近年来，CDT 在计算机科学的许多领域中得到广泛的应用，综合起来主要包括以下几个方面。

5.1 在程序语言设计中的应用研究

相对于其他的数据类型理论，由 CDT 所得到的结构化递归或共递归提供了一种简单的递归或共递归实现，且满足一系列重要的性质，如递归函数是可终止的，而共递归函数是活性的(productive)。而且，利用 fold 或 unfold 的存在性和唯一射可以保证数据类型上存在相应的递归或共递归函数，这为程序的优化、转换、推理和证明等提供了相应的理论支持。因此，这方面主要是利用 CDT 的特性来研究它们在各种程序语言中的应用。概括起来，目前 CDT 主要用于以下程序语言。

(1)函数式程序语言和 lambda 演算

Hagino 最早将 CDT 应用于函数式程序语言 CPL 和 ML，并在类型化 λ-演算中引入范畴数据构造[4,5]。随后，Wralth[6]在 Hagino 的基础上指出如何给出多态类型化 λ-演算中的 CDT 描述，从而将 CDT 中基于泛性质的描述转换为基于 λ-表达式的描述。Cockett 和 Spencer 等利用分裂纤维下强数据类型构造上的初始性和终结性给出一个范畴论组合子简化系统及相应的简化推导规则，并将其作为编程语言 Charity 中计算的抽象引擎[74]。Vene 等[23]重点探讨 CDT 上的各种递归和共递归及其计算律在 Haskell 的应用。Johann 和 Ghani 等[75]给出嵌套数据类型在 Haskell 中的应用。Capretta[76]将基于范畴论的共代数应用于函数式程序语言的类型系统研究中，而 Kurz[77]探讨了带绑定的归纳数据类型在无限 λ-演算中的应用。笔者也在文献[2]中探讨了共归纳数据类型在 Haskell 中的应用。目前许多函数式程序语言都能够支持归纳和共归纳数据类型，像 Haskell 或 ML 等还可直接定义嵌套数据类型。但大部分函数式程序语言对嵌套数据类型上各种函数定义的支持还很弱，特别是多态递归函数。

(2)逻辑编程语言

Simon 和 Mallya 等人在传统逻辑编程的基础上增加了对共归纳的支持，提出了共归纳逻辑编程[78,79]。传统基于归纳原理的逻辑编程已经广泛地应用于计算机科学中，而融合了共归纳原理的共归纳逻辑编程可以进一步提高程序语言对无限数据结构和共递归的描述及证明能力。因此，这方面还有许多值得研究的方向，如共归纳逻辑编程的共代数语义、共归纳逻辑编程在各种类型推断系统中的应用(如延迟、部分或混合静态与动态推断)以及如何对逻辑编程中 Horn 子句进行扩展以支持归纳及共归纳命题；另外，将函数式程序语言的高效率执行性和逻辑编程的灵活性结合在一起，并应用于并发性或并行性的程序建模的研究也很有意义。

(3)数据泛化编程语言

该方面主要研究如何利用 CDT 的抽象性提高程序语言

的泛型性和可重用性，特别是各种参数化条件（如基于类型或基于类型构造器）下的多态性及相应的参数化算法或程序。这方面的相关研究可见 Vesely 等[16]对基于 CDT 的参数化结合子和多态参数化函数的研究、Hinze[80]对 Generic Haskell 的研究、Gibbons[81]对泛化编程基础的研究。目前，这方面的研究已引起许多学者的关注，不过这些研究大多都是利用参数化归纳或共归纳数据类型的泛性质及相应的递归或共递归计算，而对 Mendler-风格归纳或共归纳、嵌套数据类型或依赖数据类型上的泛化编程的研究还很少。

一些学者还将函数式程序语言中的 CDT 研究扩展到其他程序语言中，如依赖类型语言（如 Alfa/AGDA）、命令式程序语言（如面向对象程序语言 Java）或并发程序语言。例如，Ancona 等[82]探讨了面向对象程序语言中的共归纳类型系统。但这方面的研究仍需继续深化和完善，例如给出面向对象中的继承、子类型和多态等概念在 CDT 上的数学解释。另外，探讨各种 CDT 在各种形式化语言（如 lambda 演算）、交互式进程或进程演算（如 CSP 或 CCS）中的应用也具有重要的意义，特别是增加对 CDT 的支持并给出相应的类型推导和转换规则。

目前，许多主流的函数式程序语言都已能够支持归纳及共归纳数据类型。不过，即使像 Charity 和 Haskell 等既支持归纳数据类型又支持共归纳数据类型的语言也都尚未很好地利用双代数将它们结合在一起。Fokkinga，Erwig 和 Nogueira 等人已经在这方面做了一些前期的探索工作。但这些研究主要是针对 CDT 的构造语义与行为语义描述，没有给出一个较为完整的类型系统及对应的推导规则，如 typing 规则、推导（reduction）规则、即时（eager）使用策略或延迟加载策略等。

5.2 在理论证明器中的应用研究

数据类型是各种自动化或交互式理论证明器的核心。由于理论证明器中经常涉及到各种无限数据结构，而传统的理论证明器在证明这些计算的活性方面存在不足，因此许多学者通过对各种理论证明器的类型理论或逻辑基础进行扩展，增加对共归纳数据类型及共归纳原理的支持，从而用于描述和证明无限数据类型的行为及性质（如活性）。这方面的研究包括 Bertot[83]对 Coq 中归纳和共归纳数据类型上的可终结性递归函数和活性共递归函数的研究、Traytel 等[30]对 HOL 中归纳和共归纳数据类型以及混合互递归（或共递归）和嵌套递归（或共递归）的研究等。

在许多学者的努力下，目前大部分的理论证明器，如 Coq，HOL/Isabelle，Agda 和 CoCASL 等，都已经开始支持共归纳数据类型。例如，Coq 就是在归纳构造演算的基础上扩展了归纳及共归纳数据类型的一个证明辅助器，能够支持基于结构化递归的可终止递归和基于防卫性的活性共递归。但仍需继续研究如何将理论证明器中的可终止性递归和活性共递归融合起来应用于模型的检测和系统的安全性及活性证明中，并增加对其它各种 CDT 的支持。

5.3 在并行计算中的应用研究

这方面以 Skillicorn[12,13,84-86]和 Banger 等[87]的研究工作为主要代表。Skillicorn 最早在文献[84]中指出 CDT 上的构造是多态的，由这些构造可自动地产生数据类型上一系列广义的映射和约简函数，且这些函数本质是并行的。Skillicorn 还指出 CDT 是抽象数据类型的扩展，且 CDT 对计算的结构化描述非常适合于并行计算[85]。Banger[87]进一步弱化了 CDT 对范畴论基础的要求，提出了 CDT 的因子分解定理。随后，Skillicorn 指出 CDT 上的操作隐藏了内部的数据表示和计算细节（如线程分解、线程与处理器间的映射关系、处理器间的通讯及同步等），可以为并行计算的复杂性提供一种抽象描述，并且可独立于具体的机器和体系结构，因此非常适合作为一种数据并行计算模型[88]；还与 Banger 进一步探讨了一些典型的 CDT（如二元树、表或树）在并行计算[13]及代价演算[86]中的应用，而 Orchard 等[89]则将 comonads 应用于并行网格编程中。这些工作大大地促进了 CDT 在数据并行计算中的研究，使得对 CDT 的研究从纯粹抽象的 lambda 演算转为更具体的类型构造。

相对于其他的数据并行计算模型，CDT 可以避免任意地选择数据并行操作，这有利于提高数据模型的并行性，而且可以为程序的推理提供代数或共代数框架。目前，这方面还存在许多问题值得继续研究。一方面，需要继续研究如何为各种 CDT 选择合适的构造方法，探讨这些构造方法之间的关系及对应的并行实现并分析 CDT 上基于同态射的转换和构造分解定理，将其扩展到几乎同态射关系上；另一方面，需要探讨其他各种 CDT 的并行计算模型及应用。目前这方面的研究都是针对归纳数据类型，那么共归纳数据类型、嵌套数据类型或者依赖数据类型是否也适合作为并行计算模型呢？如何给出共归纳数据类型上观察操作的并行分解方法和并行实现，或者嵌套数据类型上的嵌套并行性等也是很值得研究的问题。事实上，许多基于嵌套数据类型的算法往往要比基于归纳数据类型的算法更加高效[75]。另外，需要比较和分析现有的各种并行程序语言（如 NESL，SETL，Gamma 等）中的数据类型（如 lists，bag，set，tree 或 array）或并行模型（如函数式环境中基于 skeletons 的并行计算模型或命令式环境中的数据并行性）与 CDT 的差异，建立适合 CDT 的并行计算模型、并行算法、代价评估算法和方法论等。D. B. Skillicorn 等[86]已经对表和树等 CDT 上的代价评估做了一些前期的研究，但目前这方面的研究还很少。

6 总结与展望

与其他的数据类型理论相比，CDT 具有自己独特的优势。一方面，范畴论的高度抽象性、良好的扩展性和适应性使得它为数据类型的研究提供了更为高级和抽象的方法，带来了新的研究思路，并促进了范畴论在计算机科学中的应用；另一方面，CDT 可以将数据类型研究中许多长期以来被广泛研究和应用的数学基础理论有机地融合起来，特别是代数和共代数中的各种重要研究成果。

作为 CDT 中的重要组成部分，共归纳数据类型（或共代数）和归纳数据类型（或代数）构成了范畴对偶概念，因此许多学者在研究共归纳数据类型时都借鉴了归纳数据类型和代数理论的研究结果，而且许多共归纳数据类型的结果也与归纳数据类型中的结果相对应。表 1 总结了到目前为止我们所发现的共归纳数据类型与归纳数据类型中的各种概念间的对偶关系。该对偶关系是指相对应的概念在共归纳数据类型和归纳数据类型中的性质和地位对应。

表 1　共归纳与归纳数据类型之间的对偶概念

共归纳数据类型	归纳数据类型
共代数(或共代数规范)	代数(或代数规范)
终结共代数(或终结语义)	初始代数(或初始语义)
最大固定点	最小固定点
共归纳原理	归纳原理
共递归(或共迭代)	递归(迭代)
原始共递归	原始递归
Course-of-Value 共迭代	Course-of-Value 迭代
共等式逻辑(或模态逻辑)	等式逻辑
共自由构造(共自由类型)	自由构造(自由类型)
子类型	参数化
代数簇	共代数共簇

随着 CDT 研究的进一步深入，更多关于 CDT 的性质和应用可能被发现。展望 CDT 的进一步研究内容，我们认为至少包括如下几方面。

6.1　范畴数据类型的数学理论基础研究

这一方向的主要研究内容是继续完善集合范畴或完全偏序范畴上的 CDT 研究，并将其推广到其他的范畴中，特别是 Scott 域范畴、Topos 范畴、presheaf 范畴等。其中研究的问题主要包括：

(1)CDT 与基范畴及函子的关系。探讨不同范畴中的各种 CDT 与函子类型之间的关系及性质，特别是对应的代数或共代数解释，以及各种函子类型所对应的初始代数或终结共代数的存在性条件和构造方法，并从 2-范畴(或 3-范畴、n-范畴)的角度进一步研究它们的抽象关系及性质。

(2)各种 CDT 特性间的伴随关系及性质。例如，Hinze[59]证明了嵌套数据类型上许多递归模式(如带累积参数的递归、互递归或多态递归等)都可以利用一个伴随进行描述，那么对应的各种共递归模式是否也存在类似性质？

(3)寻找 CDT 上其他更复杂或更高效的递归或共递归计算模式，给出相应的性质和计算律。

(4)将目前的递归或共递归研究扩展到其他的 CDT 中，如 Mendler-风格、嵌套、依赖归纳或共归纳数据类型，并将 monads 对计算副作用和 comonads 对计算上下文环境的结构化描述与 CDT 研究结合起来。

(5)各种 CDT 的双代数结构及其性质和计算律，特别是代数构造与共代数观察函子之间存在分配律的条件、性质和函子化提升，以及初始语义和终结语语义之间的融合。

(6)比较和分析 CDT 与其他数据类型理论(如代数规范、sketch 理论、域理论或类型理论等)之间的关系及差异。

6.2　范畴数据类型的逻辑基础研究

这一方面除了继续完善代数逻辑和共代数逻辑等工作外，还需要研究共归纳数据类型与各种共代数逻辑间的关系，给出各种逻辑构造语法和组合子在 CDT 中的解释，确定适合共归纳数据类型的逻辑演算系统及共代数规范；特别是将 Kurzs，Jacobs 和 Pattinson 等将共代数模态逻辑的研究扩展并应用到 CDT 中，给出 CDT 上各种类型的共代数函子与产生合适模态逻辑之间的关系；研究代数中的等式逻辑与各种共代数逻辑之间的对偶关系，特别是进一步研究共代数共簇与代数中的等式逻辑的融合，并给出可描述 CDT 的构造语义和行为语义的逻辑系统(如双代数逻辑)。

6.3　范畴数据类型的应用研究

具体来说，目前该领域的研究主要可分为以下方面：

(1)CDT 在程序语言中的应用问题。一方面，继续研究 CDT 在函数式程序语言、逻辑编程和数据泛化(如参数化多态、泛化编程和多型)编程中(特别是在程序的设计、转换、优化和证明等方面)的应用，并将其扩展到依赖类型语言、命令式程序语言、并发程序语言和各种形式语言中；另一方面，研究基于 CDT 的类型系统，给出相应的语法、推理和转换规则。

(2)CDT 在理论证明器中的应用问题。一方面，研究如何扩展各种理论证明器的类型理论或逻辑基础，增加对各种 CDT 的支持，提高证明器对 CDT 的描述、建模及性质证明能力；另一方面，研究如何将理论证明器中的可终止性递归和活性共递归融合起来，并将其应用于模型的检测和系统的安全性及活性证明。

(3)CDT 在并行计算中的应用问题。研究各种 CDT 上的构造或观察方法之间的关系及其并行实现，给出相应的并行计算模型、并行算法、代价评估算法和方法论等，并将其与其他类型的并行计算模型进行比较和分析。

前面总结了 CDT 在程序语言、理论证明器和并行计算等方面的应用研究，但这些方面仍存在许多内容值得进一步探讨。例如，基于代数的抽象数据类型研究大大地促进了面向对象程序语言的发展，那么基于共代数的共归纳数据类型以及融合了代数和共代数的双代数上的 CDT 是否可以为面向对象程序语言带来新的研究思路和研究方法？而且，目前各种 CDT 已经广泛地应用于函数式程序语言，那么 CDT 是否也可以应用于面向对象程序语言、构件编程和其他类型的编程语言呢？

结束语　综上所述，CDT 在计算机科学中具有广阔的应用前景，可以为程序语言设计、理论证明器或并行计算等众多领域的数据类型研究提供一个基于范畴论的数学理论框架，并通过范畴论中的对偶性和伴随将计算机科学中许多非常重要的基础理论有机地融合起来。

参 考 文 献

[1] Su Jin-dian, Yu Shan-shan. Corecursion Operations and its Calculation Laws on Coinductive Data Types[J]. Journal of South China University of Technology (Natural Science Edition), 2011, 38(10): 90-95 (in Chinese)
苏锦钿，余珊珊. 共归纳数据类型上的共递归操作及其计算定律[J]. 华南理工大学学报(自然科学版)，2011，38(10)：90-95

[2] Su Jin-dian, Yu Shan-shan. Coinductive Data Types and their Applications in Programming Languages[J]. Computer Science, 2011, 38(11): 114-118 (in Chinese)
苏锦钿，余珊珊. 程序语言中的共归纳数据类型及其应用[J]. 计算机科学，2011，38(11)：114-118

[3] Hinze R. Reasoning about Codata [M]// LNCS 6299. Berlin: Springer-Verlag, 2010: 42-93

[4] Hagino T. A Categorical Programming Language[D]. University of Edinburgh, 1987: 8-56

[5] Hagino T. Codatatypes in ML[J]. Journal of Symbolic Computation, 1989, 8(6): 629-650

[6] Wralth G C. A Note on Categorical Datatypes[M]// LNCS 389. Berlin: Springer-Verlage, 1989: 118-127

[7] Malcolm G. Data Structures and Program Transformation[J]. Science of Computer Programming, 1990, 14(2/3): 255-279

[8] Meijer E, Fokkinga M, Paterson R. Functional Programming with Bananas, Lenses, Envelopes and Barbed Wire[M]// Func-

tional Programming Languages and Computer Architecture, LNCS 523. Berlin: Springer-Verlag, 1991: 215-240

[9] Geuvers H. Inductive and Coinductive Types with Iteration and Recursion[C]// Informal Proceedings of the Workshop on Types for Proofs and Programs. Bastad, Sweden, 1992: 193-217

[10] Greiner J. Programming with Inductive and Co-inductive Types [R]. Tech. Report CMU-CS-92-109, School of Computer Science, Pittsburgh: Carnegie-Mellon University, 1992

[11] Cockett J R B, Spencer D. Strong Categorical Datatypes II: A Term Logic for Categorical Programming[J]. Theoretical Computer Science, 1995, 139(1): 69-113

[12] Skillicorn D B. Structuring Data Parallelism Using Categorical Data Types[C]// Programming Models for Massively Parallel Computers. Berlin: Computer Society Press, 1993: 110-115

[13] Skillicorn D B. Parallel Implementation of Tree Skeletons[J]. Journal of Parallel and Distributed Computing, 1996, 39(2): 115-125

[14] Fokinga M M. Datatype Laws without Signatures[J]. Mathematical Structures in Computer Science, 1996, 6(1): 1-32

[15] Erwig M. Metamorphic Programming: Structured Recursion for Abstract Data Types[R]. Technical Report 242, FernUniversit at Hagen, 1998

[16] Vesely P M. Categorical Combinators for Charity[D]. The University of Calgary, 1997: 48-63

[17] Pardo A. A Calculational Approach to Strong Datatypes [R]. 8th Nordic Workshop on Programming Theory. Research Report 240, Norway, University of Oslo, 1997

[18] Pardo A. Towards Merging Recursion and Comonads[R]. Technical Report UU-CS-2000-19, University of Utrecht, 2000: 50-68

[19] Uustalu T, Vene V. Primitive (Co) Recursion and Course-of-Value (Co) Iteration, Categorically [J]. Informatica, 1999, 10 (1): 5-26

[20] Cunha M A. Recursion Patterns as Hylomorphisms [R]. Technical report DI-PURe-03. 11. 01, Department of Informatics, University of Minho, 2003

[21] Dominguez F, Pardo A. Program Fusion with Paramorphisms [C]// Mathematically Structured Functional Programming (MSFP'06). British: British Computer Society, 2006

[22] Uustalu T, Vene V. Mendler-style Inductive Types, Categorically[J]. Nordic Journal of Computing, 1999, 6(3): 343-361

[23] Vene V. Categorical Programming with Inductive and Coinductive Types[D]. University of Tartu, 2000: 63-97

[24] Cirstea C. A Coalgebraic Equational Approach to Specifying Observational Structures[J]. Theoretical Computer Science, 2002, 280(1/2): 35-68

[25] Leivant D, Ramyaa R. Implicit Complexity for Coinductive Data: a Characterization of Corecurrence[J]. Electronic Proceedings in Theoretical Computer Science, 2012, 75: 1-14

[26] Bird R, Paterson R. Generalised Folds for Nested Datatypes[J]. Formal Aspects of Computing, 1999, 11(2): 200-222

[27] Hinze R. Polytypic Functions over Nested Datatypes [J]. Discrete Mathematics and Theoretical Computer Science, 1999, 3 (4): 193-214

[28] Blampied P A. Structured Recursion for Non-uniform Data-Types[D]. University of Nottingham, 2000

[29] Abel A, Matthes R, Uustalu T. Generalized Iteration and Coiteration for Higher-Order Nested Datatypes[C]// Foundations of Software Science and Computation Structures (FoSSaCS 2003). LNCS 2620, Berlin: Springer-Verlag, 2003: 54-69

[30] Traytel D, Popescu A, Blanchette J C. Foundational, Compositional (Co) datatypes for Higher-Order Logic: Theory Applied to Theorem Proving[C]// Twenty-Seventh Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2012). 2012: 595-605

[31] Awodey S. Category Theory [M]. Oxford University Press, 2006: 265-290

[32] Gibbons J, Hutton G, Altenkirch T. When is a Function a Fold or an Unfold? [J]. Electronic Notes in Theoretical Computer Science, 2001, 44(1): 146-160

[33] Guguen J, Rosu G. Hiding More of Hidden Algebra[C]// World Congress on Formal Methods, LNCS 1709. Berlin: Springer-Verlag, 1999: 1704-1719

[34] Erwig M. Random Access to Abstract Data Types [C]// Proceedings of AMAST. 2000: 135-149

[35] Nogueira P, Moreno-Navarro J J. Bialgebra Views: A Way for Polytypic Programming to Cohabit with Data Abstraction[C]// Proceedings of the ACM SIGPLAN Workshop on Generic Programming 2008. NY: ACM, 2008: 61-73

[36] Rutten J J M M. Universal Coalgebra: a Theory of Systems[J]. Theoretical Computer Science, 2000, 249(1): 3-80

[37] Erwig M. Categorical Programming with Abstract Data Types [M]// 7th Int. Conf. on Algebraic Methodology and Software Technology, LNCS 1548. Berlin: Springer-Verlag, 1998: 406-421

[38] Fokkinga M M, Meijer E. Program Calculation Properties of Continuous Algebras[R]. Technical Report CS-R9104, CWI, Amsterdam, 1991

[39] Cockett J R, Santocanale L. Induction, Coinduction, and Adjoints [J]. Electronic Notes in Theoretical Computer Science, 2003, 69 (1): 101-119

[40] Mycroft A. Polymorphic Types Schemes and Recursive Definitions [C]// Proceedings of International Symposium on Programming. 1984: 217-228

[41] Chuang T R, Lin J L. An Algebra of Dependent Data Types [R]. Technical Report TR-IIS-06-012, Institute of Information Science, Taiwan: Academia Sinica, 2006

[42] Capretta V, Uustalu T, Vene V. Corecursive Algebras: a Study of General Corecursion[M]// 12th Brazilian Symp. on Formal Methods (SBMF 2009), LNCS 5902. Berlin: Springer-Verlag, 2009: 84-100

[43] Bonsangue M, Rot J, Ancona D, et al. A Coalgebraic Foundation for Coinductive Union Types[C]// ICALP 2014, LNCS 8573. Berlin: Springer-Verlag, 2014: 62-73

[44] Su Jin-dian, Yu Shan-shan. Subtype of Categorical Data Types [J]. Journal of South China University of Technology (Natural Science Edition), 2013, 41(9): 58-64 (in Chinese)
苏锦钿, 余珊珊. 范畴数据类型上的子类型[J]. 华南理工大学学报(自然科学版), 2013, 41(9): 58-64

[45] Miranda-Perea F E. Some Remarks on Type Systems for Course-of-Value Recursion[J]. Electronic Notes in Theoretical Computer Science, 2009, 247(1): 103-121

[46] Jeannin J B, Kozen D, Silva A. Language Constructs for Non-well- Founded Computation[C]// ESOP 2013, LNCS 7792. Ber-

lin:Springer-Verlag,2013:61-80

[47] Hinze R,Wu N,Gibbons J. Conjugate Hylomorphisms [C]// POPL'15. 2015

[48] Clouston R,Bizjak A,Grathwohl H B,et al. Programming and Reasoning with Guarded Recursion for Coinductive Types[C]// FoSSaCS 2015. 2015

[49] Su Jin-dian,Yu Shan-shan. Generalised Coiteration and its Calculational Laws[J]. Journal of South China University of Technology(Natural Science Edition),2012,40(9):62-68 (in Chinese)
苏锦钿,余珊珊. 广义共迭代及其计算律[J]. 华南理工大学学报(自然科学版),2012,40(9):62-68

[50] Su Jin-dian,Yu Shan-shan. Corecursive Operations with Parameters and the Associated Calculational Laws[J]. Journal of Computer Research and Development,2013,50(12):2672-2690 (in Chinese)
苏锦钿,余珊珊. 带参数的共递归操作及其计算定律[J]. 计算机研究与发展,2013,50(12):2672-2690

[51] Yu Shan-shan,Li Shi-xian,Su Jin-dian. Hylomorphisms with Parameters and its Associated Calculational Laws[J]. Journal of Computer Research and Development,2013,50(3):602-618 (in Chinese)
余珊珊,李师贤,苏锦钿. 一种带参数的 Hylomorphisms 及其计算律[J]. 计算机研究与发展,2012,50(3):502-618

[52] Abel A,Matthes R. (Co-)Iteration for Higher-Order Nested Datatypes[M] // Types for Proofs and Programs, Internat. Workshop,TYPES 2002,LNCS 2646. Berlin:Springer-Verlag, 2003:1-20

[53] Uustalu T,Vene V. The Recursion Scheme From the Cofree Recursive Comonad[J]. Electric Notes in Theoretical Computer Science,2011,229(5):135-157

[54] Su Jin-dian, Yu Shan-shan. Comonadic Corecursions on Strong Coinductive Datatypes[J]. Journal of South China University of Technology(Natural Science Edition),2014,42(1):128-134 (in Chinese)
苏锦钿,余珊珊. 强共归纳数据类型上的 Comonadic 共递归[J]. 华南理工大学学报(自然科学版),2014,42(1):128-134

[55] Capretta V,Uustalu T,Vene V. Recursive Coalgebras from Comonads[J]. Information and Computation,2006,204(4):437-468

[56] Pardo A. Monadic Corecursion-Definition,Fusion Laws,and Applications[C]//Coalgebraic Methods in Computer Science,Electronic Notes in Theoretical Computer Science. 1998, 11: 105-139

[57] Su Jin-dian,Yu Shan-shan. Monadic Recursions with Fixed Parameters[J]. Journal of South China University of Technology (Natural Science Edition),2014,42(7):33-39,73 (in Chinese)
苏锦钿,余珊珊. 带固定参数的 Monadic 递归[J]. 华南理工大学学报(自然科学版),2014,42(7):33-39,73

[58] Gianantonio P D,Miculan M. Unifying Recursive and Co-recursive Definitions in Sheaf Categories [C]//FoSSaCS 2004,LNCS 2987. Berlin:Springer-Verlag,2004:136-150

[59] Hinze R. Adjoint Folds and Unfolds,or:Scything Through the Thicket of Morphisms[C]// Proceedings of the 10th International Conference on Mathematics of Program Construction (MPC'10),LNCS 6120. Berlin:Springer-Verlag,2010:195-228

[60] Hinze R. Type Fusion[C]// Proceedings of the Thirteenth International Conference on Algebraic Methodology and Software Technology,LNCS 6486. Berlin:Springer-Verlag,2010:92-110

[61] Fumex C, Ghani N, Johnn P. Indexed Induction and Coinduction, Fibrationally [C] // CALCO 2011, LNCS 6859. Berlin: Springer-Verlag,2011:176-191

[62] Glimming J. Parametric (Co) Iteration vs. Primitive Direcursion [C]//CALCO 2007,LNCS 4624. Berlin:Springer-Verlag,2007: 257-278

[63] Su Jin-dian,Yu Shan-shan. Bialgebraic Structure of Abstract Data Types [J]. Journal of South China University of Technology (Natural Science Edition),2011,39(12):44-50,63 (in Chinese)
苏锦钿,余珊珊. 抽象数据类型的双代数结构[J]. 华南理工大学学报(自然科学版),2011,39(12):44-50,63

[64] Su Jin-dian, Yu Shan-shan. Bialgebraic Structures of Abstract Data Types and its Calculational Laws[J]. Journal of Computer Research and Development, 2012, 49(8): 1787-1803 (in Chinese)
苏锦钿,余珊珊. 抽象数据类型的双代数结构及其计算定律[J]. 计算机研究与发展,2012,49(8):1787-1803

[65] Bartels F. Generalised Coinduction[J]. Electronic Notes in Theoretical Computer Science,2001,44(1):67-87

[66] Cancila D, Honsell F, Lenisa M. Generailzed Coiteration Schemata [J]. Electronic Notes in Theoretical Computer Science, 2003,82(1):76-93

[67] Zhou Xiao-cong, Shu Zhong-mei. A Survey on the Coalgebraic Methods in Computer Science[J]. Journal of Software,2003,14 (10):1661-1671 (in Chinese)
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,2003,14(10):1661-1671

[68] Gumm H P. Universal Coalgebras and their Logics[J]. AJSE-Mathematics,2009,1(1):105-130

[69] Schröder L. Coalgebraic Modal Logic in CoCASL [M]// Recent Trends in Algebraic Development Techniques,LNCS 4409. Berlin:Springer-Verlag,2007:127-141

[70] Simon L. Extending Logic Programming with Coinduction[D]. University of Texas at Dallas Richardson,2007

[71] Hasuo I,Cho K,Kataoka T,et al. Coinductive Predicates and Final Sequences in a Fibration[J]. Electronic Notes in Theoretical Computer Science,2013,298(1):197-214

[72] Mossakowski T,Schrödera L,Roggenbachb M,et al. Algebraic-coalgebraic Specification in CoCASL[J]. Journal of Logic and Algebraic Programming,2006,67(1/2):146-197

[73] Klin B. Bialgebraic Methods and Modal Logic in Structural Operational Semantics[J]. Information and Computation,2009,207 (2):237-257

[74] Cockett J R B,Fukushima T. About Charity[R]. Technical Report 92/480/18,Dep. Comp. Sci. ,Univ. Calgary. 1992

[75] Johnn P,Ghani N. Haskell Programming with Nested Types:A Principled Approach[J]. Higher-Order and Symbolic Computation,2010,22(2):155-189

[76] Capretta V. Coalgebras in Functional Programming and Type Theory[J]. Theoretical Computer Science,2011,412(38):5006-5024

(下转第 39 页)

迁移至 Openstack 云平台；邮件服务也可以迁移至云平台等。

结束语 通过与已有研究工作的对比，本文提出面向最终用户和面向云服务提供商的云计算系统可用性评估方法。这两种方法都采用可用性的定义建立模型，采用虚拟机的启动时间代替 MTTR。面向最终用户的云计算系统可用性评估方法中的 MTTF 采用云服务提供商提供的参数，而面向云计算服务提供商的云计算系统可用性评估方法采用连续长时间观察收集服务器的故障种类和频率，建立统计计算模型，计算得到 MTTF。由于面向云服务器提供商的云计算系统可用性评估方法实验验证的限制，本文只对面向最终用户的云计算系统可用性评估方法进行验证。本文通过实验对私有云平台 Openstack 和公有云平台亚马逊进行可用性评估，评估结果显示该模型适合用于快速评估云计算平台的可用性，同时为用户选择云服务提供了参考建议；实验结果表明云服务器提供商的承诺达到标准，可以将一些行业关键应用服务迁移至云平台，如沃尔玛的电子商务和邮件服务等。

参 考 文 献

[1] Luo Jun-zhou, Jin Jia-hui, Song Ai-bo, et al. Cloud: computing: Architecture and Key Technology[J]. Journal on Communications, 2011, 32(7): 3-21(in Chinese)
罗军舟，金嘉晖，宋爱波，等. 云计算：体系架构与关键技术[J]. 通信学报，2011，32(7)：3-21

[2] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing [J]. Communications of the ACM, 2010, 53(4): 50-58

[3] Kim D S, Machida F, Trivedi K S. Availability modeling and analysis of a virtualized system[C]// 15th IEEE Pacific Rim International Symposium on Dependable Computing, 2009(PRDC'09). IEEE, 2009: 365-371

[4] Chuob S, Pokharel M, Park J S. Modeling and analysis of cloud computing availability based on eucalyptus platform for e-government data center[C]// 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS). IEEE, 2011: 289-296

[5] Yang Zhi-ming, Zhang Jia-hui. A quantitative calculation model of the cloud computing availability[J]. Computer Software and Applications, 2014(7): 28-29(in Chinese)
杨志明，张嘉慧. 一种云计算可用性定量计算模型[J]. 计算机软件和应用，2014(7)：28-29

[6] Che Jian-hua. Research on Performance and Availability Evaluation Methods of Virtualization Systems[D]. Hangzhou: Zhejiang University, 2010(in Chinese)
车建华. 虚拟计算系统性能和可用性评测方法研究[D]. 杭州：浙江大学，2010

[7] Ghosh R, Longo F. Scalable Analytics for IaaS Cloud Availability[J]. IEEE Transactions on Cloud Computing, 2014, 2(1): 57-70

[8] Longo F, Ghosh R, Naik V K, et al. A scalable availability model for infrastructure-as-a-service cloud[C]// 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN). IEEE, 2011: 335-346

[9] Ghosh R, Trivedi K S, Naik V K, et al. End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach[C]// 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing (PRDC). IEEE, 2010: 125-132

[10] Khazaei H, Misic J, Misic V, et al. Availability analysis of cloud computing centers[C]// 2012 IEEE Global Communications Conference (GLOBECOM). IEEE, 2012: 1957-1962

[11] Lai C D, Xie M, Poh K L, et al. A model for availability analysis of distributed software/hardware systems[J]. Information and Software Technology, 2002, 44(6): 343-350

[12] Vani B, Priya R C M. A Survey on the Security Issues in Cloud Computing[J]. International Journal of P2P Network Trends and Technology, 2014(11): 16-19

[13] Mell P, Grance T. The NIST definition of cloud computing[J]. Communication of the ACM, 2011, 53(6): 50

(上接第 18 页)

[77] Kurz A, Petrisan D, Severi P. Nominal Coalgebraic Data Types With Applications to Lambda Calculus[J]. Logical Methods in Computer Science, 2013, 9(4): 1-51

[78] Simon L, Mallya A, Bansal A, et al. Coinductive Logic Programming [M] // Etalle S, Truszczynski M. eds., ICLP 2006, LNCS4079. Berlin: Springer-Verlag, 2006: 330-344

[79] Simon L, Bansal A, Mallya A, et al. Co-Logic Programming: Extending Logic Programming with Coinduction [M] // ICALP 2007, LNCS 4596. Berlin: Springer-Verlag, 2007: 472-483

[80] Hinze R, Jeuring J. Generic Haskell: Practice and Theory[M]// Generic Programming 2003, LNCS 2793. Berlin: Springer-Verlag, 2003: 1-56

[81] Gibbons J. Datatypes-Generic Programming[M]// Spring School on Datatype-Generic Programming, LNCS 4719. Berlin: Springer-Verlag, 2007: 1-71

[82] Ancona D, Lagorio G. Coinductive Type Systems for Object- Oriented Languages [M] // ECOOP 2009, LNCS 5653. Berlin: Springer-Verlag, 2009: 2-26

[83] Bertot Y, Komendantskaya E. Inductive and Coinductive Components of Corecursive Functions in Coq[J]. Electronic Notes in Theoretical Computer Science, 2008, 203(5): 25-47

[84] Skillicorn D B. Categorical Data Types[M]// Second Workshop on Abstract Models for Parallel Computation. Oxford University Press, 1992

[85] Skillicorn D B. Questions and Answers about Categorical Data Types[R]. Technical Rreport, Queen's University, Computing and Information Science. 1994

[86] Skillicorn D B, Cai W. A Cost Calculus for Parallel Functional Programming[J]. Journal of Parallel and Distributed Computing, 1995, 28(1): 65-83

[87] Banger C R, Skillicorn D B. Constructing Categorical Data Types [R]. Research Reports, Canada: Queen's University, 1993

[88] SkilliCorn D B. Foundation of Parallel Programming[M]. Cambridge International Series on Parallel Computation, 1994

[89] Orchard D A, Bolingbroke M, Mycroft A. Ypnos: Declarative, Parallel Structured Grid Programming [C] // Proc. of DAMP 2010. NY: ACM Press, 2010: 15-24