维普资讯 http://www.cqvip.com

Curry-Howard 同构理论:研究综述

A Survey of Curry-Howard Isomorphism

张 昱 宋方敏

(软件新技术国家重点实验室 南京大学计算机科学与技术系 南京210093)

Abstract Curry-Howard isomorphism presents a correspondence between systems of formal logic and typed calculi. In this paper, first a simple system of natural deduction and a simple system of typed λ-calculus are introduced. By the description of the relation of these two systems, we define the Curry-Howard isomorphism. Then we summarize its application in programming theory and at last, we discuss in brief the necessity and future direction in this field.

Keywords Curry-Howard Isomorphism, Type theory, Proof theory

一、引言

理论计算机科学的发展吸取了大量数学和逻辑上的重要成果。逻辑是理论计算机科学十分重要的基础之一,而其中又以直觉主义逻辑对计算机科学的影响最为显著,它的思想比古典逻辑更加切合计算的观点。另一个值得一提的成果就是Alonzo Church 在1936年提出的 λ-演算系统,λ-演算所依据的是一种完全不同于逻辑的思想,它是计算机科学中函数式程序设计语言的理论基础。这是两种无论从语法上还是从语义上去观察都区别甚大的形式系统,然而它们之间却存在着某种奇妙的对应关系,这就是本文所要介绍的 Curry-Howard 同构理论(Curry-Howard Isomorphism)。

本文将首先引入两个形式系统,在直觉主义逻辑系统方面引入一个描述部分命题演算([∧,⇒]部分)的自然推理系统(Natural deduction system),而在类型演算系统方面引入一个简单的带类型 λ-演算系统(Typed λ-calculus system),通过描述这两个系统之间的同构对应关系来给出 Curry-Howard 同构的概念。在详细刻画这两个系统之前,我们首先需要对 Heyting 的思想作一些介绍。

二、Heyting 语义

通常我们对逻辑系统的解释所依据的都是指称语义的思想,即通过构造真值表的方式给出逻辑系统的语义解释,这种传统的语义解释关注的是逻辑公式(formula)的真假值,它是独立于对逻辑系统的理解、推理或曰证明之外的。而 Heyting 语义则完全是另外一种思想,它的出发点是公式的证明(proof),而不管一个公式何时为真或何时为假。Heyting 语义认为,我们通常称之为一个公式的证明的那些文字应该是这个公式本身所固有的某种属性的描述。下面给出的是命题演算的[∧,⇒]部分的 Heyting 解释:

·原子公式(atomic formula)的证明是已知的;

·公式"A∧B"的证明是一个有序对(p,q),其中 p、q 分别为公式 A 和公式 B 的证明;

·公式"A⇒B"是一个函数 f,它将公式 A 的证明 p 映射为公式 B 的证明 $f(p)$。

三、自然推理系统

我们首先引入 Gentzen 的自然推理系统来描述命题演算中的[∧,⇒]部分。

3.1 符号

我们使用 $\dfrac{\vdots}{A}$ 来表示公式 A 的一个推理(deduction),或曰 A 的一个证明。

如果从树的观点来观察,这样的一个推理可以看作是一棵有穷树,A 是这棵树的根,而树的叶子我们称之为句子,这些句子可以有两种状态:活状态和死状态。一个句子在一个推理中处于活状态,也即表示它在此推理中是活跃的,我们称这样的句子为假设。如果一个句子在推理中是不活跃的,我们则称它在这个推理中处于死状态。句子可以从活状态变换到死状态,比如通过"⇒−I"规则:

$$\frac{\begin{matrix}[A]\\ \vdots\\ B\end{matrix}}{A\Rightarrow B}\Rightarrow -I。$$

在这个规则中,从公式 B 的一个推理出发,选择其中作为假设出现的一些句子 A,我们可以构造一个新的"A⇒B"的推理,在这个过程中,在 B 的推理中起活跃作用的假设 A 被"杀"掉了,成了死状态。当然,在 B 的推理中可能有一些不是作为假设出现的 A 并没有被"杀"掉,它们仍然处于活状态。

3.2 规则

自然推理系统中的证明是通过规则来展开的,事实上我们可以把规则看作是证明的最小单位,证明中的规则是不可再分的。这里的自然推理系统使用以下一些规则:

·假设:A;

·添规则:

$$\frac{\vdots\ A\quad \vdots\ B}{A\wedge B}\wedge -I,\qquad \frac{\begin{matrix}[A]\\ \vdots\\ B\end{matrix}}{A\Rightarrow B}\Rightarrow -I;$$

·删规则:

$$\frac{A\wedge B}{A}\wedge -1E,\quad \frac{A\wedge B}{B}\wedge -2E,\quad \frac{A\quad A\Rightarrow B}{B}\Rightarrow -E。$$

其中,规则⇒−E 通常称为"假言推理"。

3.3 规则的解释

我们在这里利用前面提到的 Heyting 思想来对规则进行解释。Heyting 思想认为证明(或曰推理)是一个公式本身所固有的,根据这种思想公式可以这样来进行解释:一个公式 A 是它的所有可能的推理的集合,对于 A 的一个证明 δ,我们可以将它表示为"δ∈A"。这样,上述的规则可以理解为一种函数构造方式,即公式 A 的一个基于假设 $B_1,\cdots,B_n$ 的推理可以看作是一个函数 $t[x_1,\cdots,x_n]$,如果给函数的每个参数分别赋以值 $b_1,\cdots,b_n$(其中 $b_i\in B_i$),则得到的函数值 $t[b_1,\cdots,b_n]\in A$。上述规则的解释如下:

·仅包含单个假设 A 的推理以变量 x 来表示,x 代表 A 中的元素;

维普资讯 http://www.cqvip.com

·如果推理最终由另外两个推理通过 $\wedge$-I 规则得到，并且假设作为规则前件的两个推理分别为 $u[x_1,\cdots,x_n]$ 和 $v[x_1,\cdots,x_n]$，则该推理解释为 $\langle u[x_1,\cdots,x_n],v[x_1,\cdots,x_n]\rangle$；

·如果推理最终以 $\wedge-1E$ 规则结束，并且假设作为这一规则前件的推理为 $t[x_1,\cdots,x_n]$，则该推理解释为 $\pi^1t[x_1,\cdots,x_n]$。同样，如果推理以 $\wedge-2E$ 规则结束，则该推理应解释为 $\pi^2t[x_1,\cdots,x_n]$；

·如果推理最终以 $\Rightarrow-I$ 规则结束，以 v 表示作为这一规则前件的推理。在使用 $\Rightarrow-I$ 规则时，必然有一个假设 A 被"杀"掉了，我们以变量 x 来表示在此过程中被"杀"掉的假设 A，则 v 可以写成函数形式 $v[x,x_1,\cdots,x_n]$，这样该推理可以解释为 $\lambda x.v[x,x_1,\cdots,x_n]$，其中 x 成了约束变元；

·如果推理最终由另外两个推理通过 $\Rightarrow-E$ 规则得到，并且假设作为规则前件的推理为 $t[x_1,\cdots,x_n]$ 和 $u[x_1,\cdots,x_n]$，其中，对于给定的变量 $x_1,\cdots,x_n$，t 是一个从 A 到 B 的函数，而 u 是 A 中的元素，因此 t(u) 是 B 中的元素，也即该推理的解释为 $t[x_1,\cdots,x_n]u[x_1,\cdots,x_n]$。

四、简单的带类型 λ-演算系统

仍然回到 Heyting 思想上，如果从 Heyting 语义的角度来看待逻辑系统，我们可以做一个对应的演算系统，在这个系统中，一个公式被看作是一个类型(type)，而这个公式的证明则看成是具有该类型的项(term)。下面给出类型与项的定义。

4.1 类型

·原子类型 $T_1,\cdots,T_n$ 是类型；

·如果 U 和 V 是类型，则 $U\times V$ 和 $U\to V$ 也是类型；

·类型仅限于此。

4.2 项

·变量 $x_0^T,\cdots,x_n^T,\cdots$ 是类型 T 的项；

·如果 u 和 v 分别是类型 U 和 V 的项，那么 $\langle u,v\rangle$ 则是类型 $U\times V$ 的项；

·如果 t 是类型 $U\times V$ 的项，那么 π^1t 和 π^2t 分别是类型 U 和 V 的项；

·如果 v 是类型 V 的项，x_n^U 是具有类型 U 的一个变量，那么 $\lambda x_n^U.v$ 则是类型 $U\to V$ 的项；

·如果 t 和 u 分别是类型 $U\to V$ 和 U 的项，那么 tu 则是类型 V 的项。

4.3 项的归约和规范化

根据上面的类型和项的定义，我们引入三个关于项的等式：

$$\pi^1\langle u,v\rangle=u,\ \pi^2\langle u,v\rangle=v,\ (\lambda x^U.v)u=v[u/x]。$$

称它们为"等式"，这是一种静态的观点，或者说是一种指称语义的观点，因为等式两边的表达式所指称的是同样的对象。然而，如果从动态的角度，我们可以把这三个等式看成是三个重写规则，它们反映了一种在演算系统中常见的归约行为。

·一个项是规范(normal)的仅当这个项中不包含以下形式的子项：

$$\pi^1\langle u,v\rangle,\ \pi^2\langle u,v\rangle,\ (\lambda x^U.v)u。$$

·项 t 可以转换(convert)到项 t' 仅当 t 和 t' 分别为以下几种形式：

$t=\pi^1\langle u,v\rangle$ 且 $t'=u$，

或 $t=\pi^2\langle u,v\rangle$ 且 $t'=v$，

或 $t=(\lambda x^U.v)u$ 且 $t'=v[u/x]$。

其中项 t 称为还原项(redex)，项 t' 称为收缩项(contractum)。

·如果存在一个从 u 到 v 的转换序列：$u=t_0,t_1,\cdots,t_{n-1},t_n=v$，其中对任何 $i=0,1,\cdots,n-1$，t_{i+1} 是由 t_i 将其一个还原项替换成相应的收缩项所得到，则称项 u 可以归约(reduce)到项 v。

·如果一个存在一个规范项 u，使得项 t 可以归约到 u，则称项 u 为项 t 的规范式(normal form)。

五、Curry-Howard 同构

从上面的两个系统中我们似乎已经看到了某种对应的关系，下面我们将详细地给出关于这种对应关系的描述，也就是 Curry-Howard 同构。

5.1 证明(proof)与项(term)的对应

一个系统中的证明是无穷的，但是我们前面提到过规则可以看作是证明的最小单位，而规则是有限的，因此我们这里仅给出我们在自然推理系统中所引入的规则与 λ-演算系统中的项的对应关系，以此来代替整个证明集和项集之间的对应关系的描述。

·推理 A 对应于变量 x^A；

·推理 $\dfrac{\begin{matrix}\vdots&\vdots\\A&B\end{matrix}}{A\wedge B}\wedge-I$ 对应于项 $\langle u,v\rangle$，其中 u 和 v 分对应于推理 A 和 B；

·推理 $\dfrac{\begin{matrix}\vdots&\vdots\\A&B\end{matrix}}{A\wedge B}\wedge-1E$ 和推理 $\dfrac{\begin{matrix}\vdots&\vdots\\A&B\end{matrix}}{A\wedge B}\wedge-2E$ 分别对应于项 π^1t 和 π^2t，其中 t 对应于推理 $A\wedge B$；

·推理 $\dfrac{\begin{matrix}[A]\\\vdots\\B\end{matrix}}{A\Rightarrow B}\Rightarrow-I$ 对应于项 $\lambda x^A.v$，其中 v 对应于推理 B；

·推理 $\dfrac{\begin{matrix}\vdots&\vdots\\A&A\Rightarrow B\end{matrix}}{B}\Rightarrow-E$ 对应于项 tu，其中 t 和 u 分别对应于推理 $A\Rightarrow B$ 和 A。

5.2 关于同构

如果把5.1节中给出的关于证明与项之间的对应关系看成是两个集合之间的对应关系，那么该对应其实是一个一一映射。但是同构不仅要求两个系统具有其间存在双射关系的两个集合，它还要求这两个系统具有同样的结构，在这里即要求任何一个系统中的运算行为，都应该在另一个系统中有与之对应的运算存在。例如在 λ-演算系统中关于项有"规范化"的运算，这一概念在自然推理系统中也同样存在，我们可以这样来定义自然推理系统中"规范化"的概念：一个证明(推理)是规范(normal)的仅当其中不包含如下任何一个规则序列：

$$\frac{\begin{matrix}\vdots&\vdots\\A&B\end{matrix}}{\dfrac{A\wedge B}{A}\wedge-1E},\qquad \frac{\begin{matrix}\vdots&\vdots\\A&B\end{matrix}}{\dfrac{A\wedge B}{B}\wedge-2E},\qquad \frac{\begin{matrix}[A]\\\vdots\\B\end{matrix}}{\dfrac{A\Rightarrow B}{B}\Rightarrow-E}。$$

从这一定义可以看出，我们不需要通过 λ-演算系统来引入自然推理系统的"规范化"的概念，它可以完全独立地在自然推理系统之中进行定义。相应地，我们可以得到如下的三个等式：

$$\frac{\begin{matrix}\vdots&\vdots\\A&B\end{matrix}}{\dfrac{A\wedge B}{A}\wedge-1E}=\begin{matrix}\vdots\\A\end{matrix},\qquad \frac{\begin{matrix}\vdots&\vdots\\A&B\end{matrix}}{\dfrac{A\wedge B}{B}\wedge-2E}=\begin{matrix}\vdots\\B\end{matrix},$$

$$\frac{\begin{matrix}[A]\\\vdots\\B\end{matrix}}{\dfrac{A\Rightarrow B}{B}\Rightarrow-E}=\begin{matrix}\vdots\\B\end{matrix}。$$

维普资讯 http://www.cqvip.com

这三个等式也同样地可以看作是重写规则，这样即可在自然推理系统中引入“转换”、“归约”、“规范”这些我们已经在λ-演算系统中定义过的概念，这些概念同样地可以在自然推理系统中独立地进行定义而不需要事先引入λ-演算系统。除了概念之外，两个系统中的一些定理之间也具有同样的对应关系，例如λ-演算系统中的范式定理(Normal form theorem)、合流定理(Church-Rosser property)在自然推理系统中均有对应的定理存在，具体内容读者可以参考文[6]与[11]。

5.3 其他的 Curry-Howard 同构

Curry-Howard 同构不仅限于以上两个系统之间的对应，它揭示了不同层次的形式系统之间的对应，例如最小命题演算对应于简单的带类型λ-演算，一阶谓词逻辑对应于依赖类型系统(Dependent types)，二阶谓词逻辑对应于多态类型系统(Polymorphic types)等等，更为完整的论述请参考文[11]。

六、研究概况

6.1 Curry-Howard 同构理论在程序设计领域中的延伸

至此，我们对 Curry-Howard 同构的介绍都仅限于纯粹的形式系统，然而通常在计算机科学中，我们所希望的不仅仅是数学形式上的一种完美的理论，而更需要一种能够在计算机应用领域中具有指导意义的理论，Curry-Howard 同构理论也正满足了这一要求。

确切地讲，Curry-Howard 同构实际上描述了在证明理论与类型理论之间的对应关系，从前面的介绍我们也可以看出这一点，它的起始点正是证明与类型之间的对应。这两种理论相比起来，类型理论似乎更容易延伸到程序设计领域，因为带类型的演算系统更能反映计算的观点，一个项的类型精确地刻画了这个项的计算的目的。从程序设计理论的观点来观察类型，一个类型 A 可以看作是一个程序 t 的规格说明(Specification，也即一般意义上的关于程序 t 的目的的描述或注释，只不过在这里我们应该用一种更为精确的形式语言来表述)，而相应的程序 t 则视为具有该类型的项，显然程序执行的目的是为了达到规格说明中所提出的要求，并且满足同一个规格说明的程序不止一个。依据这样的观点，一个带类型的演算系统则可视为一个高级程序设计语言，而无类型的演算系统则可视为低级的程序设计语言，比如机器语言。低级语言的程序之中是没有注释的，也不具备可读性，但是这对计算机是没有影响的，因为所有的注释对计算机都是无意义的。例如，无类型的λ-演算即可视为一个机器语言的实现，只需对其中的一些基本符号给予硬件上的定义。

证明论与程序设计理论的关系似乎并不十分明显，但是因为有了 Curry-Howard 同构理论，根据前面类型论在程序设计领域中的延伸，我们同样可以将证明论也延伸到程序设计领域中去，这就是近年来的一个新兴的研究方向：基于证明的程序设计(Programming with proofs)。这种程序设计方法是建立在二阶逻辑的基础之上的，它将程序的规格说明看成是谓词公式，根据这些公式的证明来导出相应的程序，具体的论述可以参考文[9]。目前已经有不少逻辑与计算机科学领域的学者在这方面做出了一些成果，在基于直觉主义证明的程序设计方面，Jean-Yves Girard 的 F 系统与 Jean-Louis Krivine 的 FA2系统比较具有代表性，它们分别是基于二阶命题逻辑和二阶谓词逻辑所构作的。其中 F 系统曾分别从逻辑领域和计算机领域单独提出，在逻辑领域，Girard 为了给他在二阶逻辑上的研究设计一套证明表示体系而构造出这一系统，而在计算机领域，John Reynolds 则在试图为多态程序设计语言构造一个类型系统时构造了同样的一个系统，这也正体现了逻辑系统与程序设计理论之间的对应关系。F 系统的详细论述可参考文[5]、[6]，FA2系统的详细论述可参考文[8]。

6.2 研究的历史与必要性

我们可以看出，Curry-Howard 同构实际上是 Heyting 语义在语法层面的反映，它的思想仍要归属于直觉主义逻辑思想。Curry 在文[4]中发现，当以蕴涵(implycation)来标识函数类型时，最小命题演算的 Hilbert 表示中的可证公式与组合逻辑中的类属类型是一致的，而且逻辑中的证明对应于函数演算中的项，反之亦然。他还发现最小命题演算的自然推理表示与简单的类型λ-演算之间也存在有同样的对应关系。而关于证明的规范化与项的归约之间的关系，则由 Howard 于1968年在文[7]中给出，他指出最小命题逻辑的自然推理系统中证明的归约应该与简单的类型λ-演算系统中相应项的β-归约对应。

其后，随着多种类型λ-演算系统与相应的自然推理逻辑系统的产生，Curry-Howard 同构也得到了进一步的发展，并逐渐延伸到计算机科学领域，特别是在程序设计理论方面产生了十分重要的影响，提出了证明理论与程序设计理论之间的对应，这是 Curry-Howard 同构理论最为重要的应用，在6.1节中我们已经有所论述。除此之外，Curry-Howard 同构在定理的自动证明方面也具有十分重要的指导意义。

结束语 Curry-Howard 同构理论揭示了证明理论、类型理论与程序设计理论之间潜在的对应关系，并且在语法层面给出了严格的同构关系的表述，在逻辑学领域和计算机科学领域都产生了十分重要的影响。这些影响不仅表现在因此产生的一些新的研究课题上，如基于证明的程序设计方法等，也表现在对研究方法的指导上。例如在逻辑方面，如果独立地构造出一个逻辑证明系统或是一个类型演算系统，我们就应该考虑构造一个相应的类型系统或证明系统，以完善对我们所构造的系统的讨论。另外，在计算机科学方面，如何将证明理论和类型理论更好地应用到程序设计理论中，特别是如何利用证明理论来开发程序与验证程序的正确性，这些问题仍将是这一领域今后研究的主要方向。

参 考 文 献

1 Barendregt H P. Lambda Calculi with Types. Handbook of Logic in Computer Science. Oxford University Press，1992，2：117～309

2 Barendregt H P. The impact of the lambda calculus in logic and computer science. Bulletin of Symbolic Logic，1997，3(2)：181～215

3 Barendregt H P. The Lambda Calculus：Its Syntax and Semantics. North-Holland，second，revised editin，1984

4 Curry H B，Feys R. Combinatory Logic. North-Holland，1958

5 Girard J Y. The system F of variable types，fifteen years later. Theoretical Computer Science，1986，45：159～192

6 Girard J Y，Lafont L，Taylor P. Proofs and Type. Cambridge Tracts in Computer Science 7，Cambridge Univ. Press，1989

7 Howard W. The formulae-as-types notation of construction. In Seldin & Hindley[10]，479～490

8 Krivine J L. Lambda-calculus，types and models. Ellis-Horwood，1993

9 Krivine J L，Parigot M. Programming with proofs. J. Inform. Process. Cybernet，1990，3：149～167

10 Seldin J P，Hindley J R，eds. To H. B. Curry：Essays on Combinatory Logic，Lambda Calculus and Formalism. Academic Press Limited，1980

11 Sørensen M H B，Urzyczyn P. Lectures on the Curry-Howard Isomorphism：[Technical Report 98/14]. DIKU，Copenhagen，1998