何幻

Programming is about ideas,

languages are just a way to express them.

语言背后的代数学（五）：Σ代数

2018-02-03 | Math

回顾

上文我们介绍了哥德尔定理，它指出了形式化方法的局限性，
任何包含初等算术Π的形式理论，都是不完全的，
且自身的协调性无法在系统内部被证明。

为了理解这句话，上文中我们做出了严谨的定义，
仔细建立了语法和语义之间的联系。

实际上，语法（符号）层面的推导，属于公式的证明，
而语义（模型）层面的推导，属于逻辑结论的推理。
证明和推理之间的关系由系统的可靠性和完全性给出。

1. 简单类型化λ演算

在《你好，类型》系列文章中，

我们介绍了简单类型化λ演算（simply typed lambda calculus）$\lambda^{\rightarrow}$，

它是一个形式系统，采用公理化的方式定义。

当时我们看来，系统中的λ项，只是一堆符合推导规则的符号，

我们并不知道它到底代表什么含义。

例如，$\lambda x:T.x+1$，只是一个符号串，

自然数集上的后继函数 f(x)=x+1 ，能不能作为它的解释，我们是不清楚的，

只是猜想可能是。

不幸的是，后继函数并**不足以**作为$\lambda x:T.x+1$的解释，

因为，集合上的后继函数是没有不动点的，而$\lambda x:T.x+1$有不动点$\perp$。

我们曾经在《递归函数》系列文章中给出过证明。

2. Σ代数

一般有两种通用的方法，来给出简单类型化λ演算$\lambda^{\rightarrow}$的语义，

一种是**Henkin模型**，另一种是笛卡尔闭范畴。

范畴论我们可以稍后再介绍，这里先介绍Henkin模型，

不过在这之前，我们还得先了解一些Σ代数相关的内容。

Σ代数是一种数学结构，

一个Σ代数，包含了一个或多个集合，称为**载体**（carrier），

以及一些特征元素，和载体上的一些一阶函数，

$f : A_1 \times \cdots \times A_k \to A$

例如，Σ代数$\mathscr{N} = \langle N, 0, 1, +, \cdot \rangle$

具有载体N，它是自然数集，

具有特征元素，$0, 1 \in N$，

以及函数，$+, * : N \times N \to N$。

其中，特征元素可以看成零元函数。

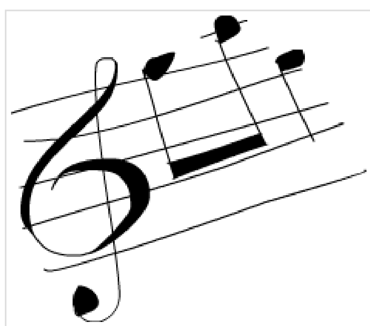
带有多个载体的例子是Σ代数

$\mathscr{A}_{pcf} = \langle N, B, 0, 1, \cdots, +, true, false, Eq?, \cdots, \rangle$

其中N是自然数集，B是布尔值集，

$0, 1, \cdots$是自然数，$+$是加法函数。

3. 代数数据类型的签名（signature）



在简单类型化λ演算$\lambda^{\rightarrow}$中，类型属于形式系统中的概念，

它并不代表类型中值的集合。

这种认识可能有助于澄清人们对编程语言中类型的误解。

例如，我们可以为初等算术系统Π赋予类型，

指定$0 : nat$，$1 : nat$，$+ : nat \times nat \to nat$，$\cdot : nat \times nat \to nat$，

分别为常元符号0和1，以及二元函数符号$+$和$\cdot$的类型。

常元符号也可以看成是**零元**函数符号。

这里，我们称以下二元组$\langle S, F \rangle$，

为初等算术系统Π的类型签名。

其中，S是系统中类型的集合$\{nat\}$，
F是函数符号的集合$\{0:nat, 1:nat, +:nat\times nat\rightarrow nat, \cdot:nat\times nat\rightarrow nat\}$。

一般的，一个类型签名（signature）$\Sigma=\langle S,F\rangle$，由以下两部分构成，
（1）S是以类型为元素构成的集合，
（2）F是类型上函数符号的集合，$F=\{f:s_1\times\cdots\times s_k\rightarrow s\}$
其中，$s_1,\cdots,s_k,s\in S$。

并且，除了初等算术系统Π，某些系统中可能还会包含变量，
因此，为了完成类型化，我们还需为这些变量指定类型。

我们称有限集$\Gamma=\{x_1:s_1,\cdots,x_k:s_k\}$，
为变量$x_1,\cdots,x_k$的一个**指派**（assignment）。
其中，$s_1,\cdots,x_k$是类型。

有了签名和指派之后，
类型为s的项的集合$Terms^s\left(\Sigma,\Gamma\right)$就可以这样定义了，
（1）如果$x:s\in\Gamma$则$x\in Terms^s\left(\Sigma,\Gamma\right)$
（2）如果$f:s_1\times\cdots\times s_k\rightarrow s$且$M_i\in Terms^{si}\left(\Sigma,\Gamma\right)$，$i=1,\cdots,n$，
则$fM_1\cdots M_k\in Terms^s\left(\Sigma,\Gamma\right)$

具有多种类型的项的集合可以记为$\{Terms^s\left(\Sigma,\Gamma\right)\}_{s\in S}$，
其中S为类型的集合。

4. 项的解释

Σ代数，与类型化的项的集合之间，存在着解释关系。

如果满足以下两个条件，

（1）对于每一个类型$s \in S$，恰好存在Σ代数中的一个载体A^s与之对应，

（2）每一个函数符号$f : s_1 \times \cdots \times s_k \to s$，

恰好存在集合上的一个函数$\mathscr{I}(f) : A^{s1} \times \cdots \times A^{sk} \to A^s$与之对应，

$\mathscr{I}(f)$也可以写成$f^{\mathscr{A}}$。

我们就称$\mathscr{A} = \left\langle \{A^s\}_{s \in S}, \mathscr{I} \right\rangle$就是$\{Terms^s(\Sigma, \Gamma)\}_{s \in S}$所对应的$\Sigma$代数。

为了解释含变量的类型化的项，我们需要定义**环境**的概念。

Σ代数$\mathscr{A}$的环境η，指的是把变量映射到$\mathscr{A}$的各载体中元素的一个映射，

$\eta : \mathscr{V} \to \cup_s A^s$

对于含变量x的项M，η为它指定了载体上的一个唯一确定的值。

如果对于指派Γ而言，$\forall x : s \in \Gamma$，都有$\eta(x) \in A^s$，

我们就说环境η满足指派Γ。

假定Σ代数$\mathscr{A}$的一个环境η满足指派Γ，

在这个环境中，我们就可以将任何项$M \in Terms(\Sigma, \Gamma)$的含义$\mathscr{A}[\![M]\!]\eta$ 定义如下，

（1）$\mathscr{A}[\![x]\!]\eta = \eta(x)$

（2）$\mathscr{A}[\![fM_1 \cdots M_k]\!]\eta = f^{\mathscr{A}}(\mathscr{A}[\![M_1]\!]\eta, \cdots, \mathscr{A}[\![M_k]\!]\eta)$

5. 例子

上文我们介绍了初等算术系统Π的类型签名$\langle S, F \rangle$，

其中，$S = \{nat\}$，$F = \{0 : nat, 1 : nat, + : nat \times nat \to nat, \cdot : nat \times nat \to nat\}$。

我们可以选择Σ代数$\mathscr{N} = \langle N, 0, 1, +, \cdot \rangle$作为它的解释，

它的载体为自然数集N，$0, 1, +, \cdot$分别为自然数集上的零元和一元函数。

如果初等算术系统Π中的项包含变量，
我们就可以为Σ代数$\mathscr{N}$指定环境η。

例如，我们可以假定环境η满足$\eta(x)=0$，
则在这个环境中，$x+1$的语义就可以按下式确定了。
$[\![x+1]\!]\eta=+^{\mathscr{N}}([\![x]\!]\eta,[\![1]\!])\eta=+^{\mathscr{N}}(\eta(x),1^{\mathscr{N}})=+^{\mathscr{N}}(0^{\mathscr{N}},1^{\mathscr{N}})=1$

总结

本文介绍了一种称为Σ代数的数学结构，
它可以用来解释带有类型签名的项。

可是，要想让这样的Σ代数称为$\lambda^{\rightarrow}$项的模型，还是不够的，
我们还必须保证每一个$\lambda^{\rightarrow}$项的解释，都在模型中。
为此Σ代数还要满足一些额外的条件。

下文我们再详细讨论这些条件。

参考

你好，类型（六）：Simply typed lambda calculus
递归函数（九）：最小不动点定理
程序设计语言理论基础

语言背后的代数学（四）：哥德尔定理
语言背后的代数学（六）：Henkin模型

© 2018
由 Hexo 强力驱动 | 主题 - NexT.Pisces