

Type Theory And Formal Proof

3. Second order typed lambda calculus 二階有型別 lambda 運算

3.1 型別抽象與型別套用

$x \rightarrow \lambda x.M$ ， M 中所有裡面的 x 變成 bound variable（約束變數）。

$\lambda x : \sigma : M$ 取決於(depends on)term $x \Rightarrow \lambda_{\rightarrow}$ 可以 terms depends on the terms

MN 是型別的應用

如此可說是一階(first order)抽象

本章討論 terms depend on types (second order operation/second order dependency) 可以理解成依型別值（？）

本章討論 λ_2 ：二階具型別 lambda 演算

範例： $\text{id}(x) = x$ 函數

若是自然數，則有 $\lambda x : \text{nat}. x : \text{nat} \rightarrow \text{nat}$

若是布林值，則有 $\lambda x : \text{bool}. x : \text{bool} \rightarrow \text{bool}$

還有函數 $\lambda x : (\text{nat} \rightarrow \text{bool}). x : (\text{nat} \rightarrow \text{bool}) \rightarrow (\text{nat} \rightarrow \text{bool})$

可是這樣函數要每個型別就重複定義一次，有無更好的方法？

我們需要定義一個任意型別 α ，變成： $f \equiv \lambda x : \alpha. x$

但是對於 $M \in \text{nat}$ ， $f M$ 是 invalid，因為 $\alpha \neq \text{nat}$ 。所以要進行抽象化：

$\lambda \alpha : *. \lambda x : \alpha. x$

其中 α 是 type， $*$ 是 kind。

所以

$(\lambda \alpha : *. \lambda x : \alpha. x) \text{nat}$

$\xrightarrow[\beta]{} \lambda x : \text{nat}. x$

且 $(\lambda \alpha : *. \lambda x : \alpha. x) (\text{bool} \rightarrow \text{nat})$

$\xrightarrow[\beta]{} \lambda x : (\text{bool} \rightarrow \text{nat}). x$

然而我們需要 β -reduction