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

$x \to \lambda x.M$，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 => $\lambda_{\to}$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$\lambda 2$：二階具型別 lambda 演算

範例：$\mathrm{id}(x) = x$函數

若是自然數，則有$\lambda x:nat.\ x:nat \to nat$

若是布林值，則有$\lambda x:bool\ .\ x:bool \to bool$

還有函數$\lambda x:(nat \to bool).\ x:(nat \to bool) \to (nat \to bool)$

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$M \in nat$，$f\ M$是 invalid，因為$\alpha \not\equiv nat$。所以要進行抽象化：

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

其中α 是 type，$*$是 kind。

所以

$(\lambda\alpha:* .\lambda x:\alpha.x)nat$

$\underset{\beta}{\twoheadrightarrow} \lambda x:nat.\ x$

且 $(\lambda\alpha:* .\lambda x:\alpha.x)(bool \to nat)$

$\underset{\beta}{\twoheadrightarrow} \lambda x:(bool \to nat).\ x$

然而我們需要β-reduction，之後介紹。

第二個範例，就是關於 iteration 迭代。

設有型別σ，和函數$F:\sigma \to \sigma$，定義$D_{\sigma,F}$是函數，$\lambda x:\sigma.F(F(x))$。所以$D_{\sigma,F}$是F的第二迭代，也標記為$D \circ D$。我們如何讓σ不是指單一個型別，而是任意型別的代稱呢？我們又可以用什麼方法讓F可以是任意函數呢？我們可以用下列表示方式：

$D \equiv \lambda\alpha:* .\lambda f:\alpha \to \alpha.\ \lambda x:a f(fx)$，$D$叫做多型 polymorphic 函數，第一個抽象 abstration 是二階 second-order 的。因此這樣：

$D\ nat \underset{\beta}{\twoheadrightarrow} \lambda f:nat \to nat\ .\lambda x:nat\ f(fx)$其中 $\lambda x.f(fx)$即$f \circ f$

另外$D\ nat\ s \underset{\beta}{\twoheadrightarrow} \lambda s.\ s(sx)$這是函數將自然數$x$轉成$s(sx)$，亦即「`(x){x+2}`」。至於表示函數合成的$x \circ y$要如何表示？

我們可以用下列表示方式： $\circ \equiv \lambda\alpha : * .\lambda\beta : * .\lambda\gamma : * .\lambda f : \alpha \to \beta.\lambda g : \beta \to \gamma.\lambda x.g(fx)$

因此$\circ\ A\ B\ C\ F\ G$可以 beta-reduction，轉為$G \circ F$，即$\lambda x : A.G(F(x))$。