Type Theory And Formal Proof

第 2 章：簡單型別 lambda 運算(simple typed lambda calculus)

2.2 simple type 簡單型別

型別變數 type variable：$\mathbb{V} = \{\alpha, \beta, \gamma, ...\}$用希臘字母表示。

$\mathbb{T}$：所有簡單型別，定義如下：

1. 型別變數：$\alpha \in \mathbb{V} \Rightarrow \alpha \in \mathbb{T}$，表達基本型別，比如 list, nat 等
2. 箭頭型別：$\alpha, \tau \in \mathbb{T} \Rightarrow (\alpha \to \tau) \in \mathbb{T}$

箭頭→是右結合的，和函數的 apply 代入不同。比較$\alpha \to \beta \to \gamma = (\alpha \to (\beta \to \gamma))$和$x_1 x_2 x_3 = ((x_1 x_2) x_3)$。

註：在本書中，$\mathbb{N}$ 和 $\mathbb{L}$ 指數學世界的自然數和列表，而 nat 和 list 指電腦程式世界的同樣的型別。

「term M有類型（type、型別）σ」寫成$M : \sigma$

type 有唯一性。比如：若$x : \sigma$且$x : \tau$，則$\sigma \equiv \tau$

簡單型別 lambda 演算的出現的推演規則：

1. application（代入）：若 $M : \sigma \to \tau$ 且 $N : \sigma$，則 $MN : \tau$
2. abstration（抽象）：若 $x : \sigma$ 且 $M : \tau$，則 $\lambda x.M : \sigma \to \tau$

$(x\ x)$在這種情況下，因為不可能既是$x : \alpha \to \beta$且$x : \alpha$這種型別存在，所以這種式子不會被構造到。

若$\exists \sigma$ 滿足 $M : \sigma$，則M是可賦予型別的(typable)。

2.3 Church-typing (explicit typing) 和 Curry-typing (implicit typing)

1. typing à la Church（explicit typing，外顯型別）：先給定型別予變數，再推出其他表達式的型別。
2. typing à la Curry（implicit typing，隱藏型別）：先給定一個表達式，再推論其內變數可能的型別。

explicit typing 的案例：

設$M \equiv ((\lambda x.\lambda y.x)(u\ v))$，如果$v : \alpha \to \alpha$，$u : (\alpha \to \alpha) \to \beta$，$x : \beta$，$y : \gamma$，則$M : \gamma \to \beta$

implicit typing 的案例（需要用推理和類似合一 (unification) 的方法）： $M \equiv ((\lambda x.\lambda y.x)(u\ v))$，可以推論：

$v : \alpha$

$u : \alpha \to \beta$

$\lambda x.\lambda y.x : \gamma \to \delta$

$x : \gamma$

$\lambda y.x : \delta = \varepsilon \to \zeta$

$y : \varepsilon$

$x : \gamma = \zeta$

$\lambda y.x : \delta = \varepsilon \to \gamma$

$\lambda x.\lambda y.x : \gamma \to \varepsilon \to \gamma$

$(u\ v) : \beta = \gamma$

$u : \alpha \to \gamma$

$M : \varepsilon \to \gamma$

但是 implicit typing 的型別變數，只是一種示例，可以把β用「$\omega \to \omega$」這種形式取代。

本書常用 explicit typing。

我們用上面的 explicit typing 的範例，

$u : (\alpha \to \alpha) \to \beta, v : (\alpha \to \alpha)$ 可以推論到

$((\lambda x.\lambda y.x)(u\ v)) : \beta \to \gamma$，

則可以寫成：

在上下文$u : (\alpha \to \alpha) \to \beta, v : (\alpha \to \alpha)$下，$((\lambda x.\lambda y.x)(u\ v)) : \beta \to \gamma$

用形式語言的方式寫出來如下： $u : (\alpha \to \alpha) \to \beta, v : (\alpha \to \alpha) \vdash ((\lambda x.\lambda y.x)(u\ v)) : \beta \to \gamma$

$\vdash$ 表示左邊的可以推論到右邊的能力(derivibility)。(註：可以參考弗雷格的著作)

2.4 Church lambda→演算的推演規則 (derivation rules)

先賦予型別的 lambda term，其名為$\Lambda_{\{\mathbb{T}\}}$，定義如下:

$\Lambda_{\mathbb{T}} = V|(\Lambda_{\mathbb{T}}\ \Lambda_{\mathbb{T}})|(\lambda V : \mathbb{T}.\Lambda_{\mathbb{T}})$，其中$V$表變數的集合。

定義

1. statement 形如$M : \sigma$，其中$M \in \Lambda_{\mathbb{T}}$且$\sigma \in \mathbb{T}$（$\sigma$是型別）。$M$稱為主體(subject)，$\sigma$稱為類型(type)。
2. declaration（宣告）是有變數當主體的 statement
3. context（上下文）是一系列不同主體（不同變數）的宣告列表（註：context 可為空）
4. judgement（判斷）形如$\Gamma \vdash M : \sigma$，其中左邊的Γ是上下文，右邊的$M : \sigma$是 statement

Premiss 前提和 Conclusion 表達式：

$\gamma \vdash x : \sigma$ 若 $x : \sigma \in \Gamma$

$\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \tau\ \Gamma \vdash M\ N : \tau$

$\Gamma, x : \sigma \vdash M : \tau$

$\Gamma \vdash \lambda x : \sigma M : \sigma \to \tau$