「教堂」的蘭達

文／林一平 講座教授

林一平手繪之邱奇 (Alonzo Church)。林一平提供

我實作物聯網平台 IoTtalk，設計一個機制，很適合執行蘭達函式 (λ-function)。對於「蘭達」，我一直情有獨鍾。

在 1986 年 9 月，我在西雅圖的華盛頓大學接觸到第一堂計算機理論課。授課教授 Paul Yang 一直提到「教堂的蘭達、圖靈的機」，彷彿是江湖黑話，幫派切口。左顧右盼，四周的白人同學聽得津津有味，談笑有鴻儒，頻頻點頭，我則是一頭霧水，猶如雞立鶴群之白丁，也不敢發問，怕鬧笑話。

下課後衝到圖書館翻書，想搞懂啥是教堂的蘭達。最後搞懂，教堂 (Church) 是人名，翻譯為「邱奇」。邱奇 (Alonzo Church) 是數學家，也是圖靈 (Alan Mathison Turing) 的指導教授。1930 年，邱奇以數學邏輯為基礎，提出蘭達演算法 (λ-calculus)，以變數綁定和替換的規則，發展出基於函式 (function) 以及遞迴機制的形式系統。λ-calculus 是一個通用的計算模型，強調函式變換規則的運用，而非實現它們的具體機器。

邱奇的學生圖靈也發表了一個簡單形式的抽象裝置，後人稱為「圖靈機」(Turing Machine)，是能力和蘭達演算法能力相同的計算模型。圖靈機的執行龜速，沒有實際用途，但引導後人想像，這種機器能解決運算問題。

圖靈嚴謹的證明出，我們不可能用一個演算法來決定一部指定的圖靈機是否會停機。邱奇在 λ-calculus 方面相等的證明比圖靈的發表早了幾個月 (老師還是比較厲害)。不過圖靈的做法比邱奇直觀，更易於理解。

當年被認為不直觀的 λ-calculus 衍生出 λ-function，反而實際應用於現代程式語言如 Java、C# 及 Python。原因是 λ-calculus 清晰地定義何謂「可計算函式」。任何可計算函式都能以 λ-function 表達並據以求值，相當於單一磁帶圖靈機的計算過程。λ-function 是匿名函式，不需要定義名稱，只有一行運算式，語法非常簡潔，功能強大，適用於小型的運算。

我設計的 IoTtalk 物聯網平台以 Python 實作，並以圖形化使用者介面將物聯網設備 (感測器與制動器) 以圖符 (icon) 形式呈現。IoT 圖符間可連線，我在連線間畫了一個小圈圈。當初設計這個小圈圈，是一個產生 λ-function 的機制，藉此寫出優雅的函式。只可惜部分 IoTtalk 使用者不曉得我的苦心，老是寫馮紐曼 (von Neumann) 架構的駑鈍函式，我的俏媚眼做給瞎子看，只能徒呼負負。

林一平
國立陽明交通大學資工系終身講座教授暨華邦電子講座

現為國立陽明交通大學資工系終身講座 教授暨華邦電子講座，曾任科技部次長，為 ACM Fellow、IEEE Fellow、AAAS Fellow 及 IET Fellow。研究興趣為物聯網、行動計算及系統模擬，發展出一套物聯網系統 IoTtalk，廣泛應用於智慧農業、智慧教育、智慧校園等領域 / 場域。興趣多元，喜好藝術、繪畫、寫作，遨遊於科技與人文間自得其樂，著有 < 閃文集 >、< 大橋驟雨 >。

Church's Lambda

While implementing an IoT platform called IoTtalk, I once designed a mechanism most suitable for executing a lambda function (λ-function). I have always had a soft spot for “lambda”.

I took my first computer theory course at the University of Washington in Seattle in September 1986. Professor Paul Yang always mentioned "Church's lambda, Turing's machine" during class. These terms seemed like gangster slang or gang signs to me. When I looked around, my white classmates sitting nearby listened with relish, nodded frequently, and laughed and chattered as if they actually understood. I, on the other hand, was totally lost and felt like an insignificant chicken standing among a flock of superior cranes. I was afraid to ask questions for fear of being laughed at by others.

Rushing into the library right after class, I flipped through books to figure out what Church's lambda was. At last, I realized that Church was the surname of Alonzo Church, a mathematician and the advisor to Alan Mathison Turing. In 1930, on the basis of mathematical logic, Church proposed the λ-calculus, which defined the rules of variable binding and substitution to develop a formal system based on functions and recursion. The λ-calculus is a general computational model that emphasizes the usage of functional transformation rules rather than the specific machines that implement them.

Church's student Turing also introduced a simple abstract machine, later known as the "Turing machine," that expresses another computational model equivalent in power to the lambda algorithm. A Turing machine is not a computation model for practical problems because of its extremely low execution efficiency. Nevertheless, it inspired future generations to imagine that such devices could solve computing problems.

Turing proves rigorously that there is no algorithm that will determine whether a given Turing machine will halt. Even though Church's equivalence proof in the λ-calculus was published a few months earlier than Turing, Turing's approach is much more intuitive, and easier for readers to understand.

Although the λ-calculus was considered unintuitive at the time, a λ-function, derived from the λ-calculus, is widely used in modern programming languages such as Java, C#, and Python. The λ-calculus clearly defines what a "computable function" is. Any computable function can be expressed and evaluated using a λ-function, which is equivalent to the calculation process of a single-tape Turing machine. A λ-function is an anonymous function with no need for a user-defined name. A single-line expression of a λ-function with concise syntax can carry out powerful functionality. Therefore, a λ-function is quite suitable for simple logical operations.

Being implemented in Python, the IoTtalk I developed presented IoT devices (sensors and actuators) as icons with a graphical user interface. These IoT icons could be connected, and I placed small clickable circles on the links between icons. The original design of this small circle was to provide a mechanism to embed a λ-function so as to generate elegant functions. It was a pity that some IoTtalk users did not even know my ingenuity and always wrote blunt functions based on von Neumann's architecture. All my efforts were in vain, just as a smiley beauty winked at a blind. I could do nothing but murmur, “What a waste!”

Dr. Jason Yi-Bing Lin

Lifetime Chair Professor of the Department of Computer Science at National Yang Ming Chiao Tung University and Winbond Chair Professor

Dr. Lin is currently a lifetime chair professor of the Department of Computer Science at National Yang Ming Chiao Tung University and Winbond chair professor. He is an ACM Fellow, IEEE Fellow, AAAS Fellow and IET Fellow. His research interests include Internet of Things, mobile computing, and system simulation. He has developed an Internet of Things system called IoTtalk, which is widely used in smart agriculture, smart education, smart campus, and other fields. He has a variety of interests, such as art, painting, and writing, as well as voyaging through science, technology, and humanities.