关于“思考函数式编程”几点注记

■ 文 / 裘宗燕

看了2008年第7期蔡学镛先生的“思考函数式编程”，我觉得有几个问题需要澄清。

第一个问题最简单，那就是术语。“formal system”最好还是用“形式系统”，这一术语已被数学界、逻辑界和计算机科学界广泛使用。计算机学术领域有许多人研究“formal methods”，形式化方法，软件工程里有“形式化开发”（formal development），逻辑中有“形式证明”（formal proof）。说是“正规开发”、“正规证明”可能要遭大家骂，“我们做的开发不正规？”第二个术语是“calculus”，已被广泛译为“演算”。“λ演算”是一例，另外有“命题演算”、“谓词演算”等不胜枚举。把calculus叫做“算术”，那么arithmetic怎么办？

第二个问题牵涉到历史。1930年代还没有computation（计算）的概念，Church、Turing（有趣得很，他当时是Church的研究生，他们并不是独立研究）等人希望弄清楚的是那时称为“能行过程”（feasible procedure）的现象。所谓“能行过程”是人们的一种直观看法，指规定了一套动作规则，机械地按规则执行的过程性行为。Church和Turing想从理论上定义清楚什么是能行过程，能行过程究竟能做什么。后来他们在工作中的重要贡献是证明了能行过程不能做什么。正是Church和Turing等人的工作建立起了计算（computation）这一重要概念。在这方面做出重要贡献的另两位是Kleene（也是Church的研究生）和Post，通常人们认为这四位大师是计算领域的开拓者。Gödel也做了些工作，但人们一般不把他看作计算领域的先驱，因为他主要研究逻辑和证明。文中提到的另一个著名人物von Neumann当时正忙于别的数学领域。到了1946年（十年多以后），他在某个偶然机会得知有人正在造计算机，觉得非常有趣就跑来凑热闹，当然是凑得极其不同凡响。虽然von Neumann是传奇式人物，但他对计算概念的创建和发展并没有深刻的理论贡献。“存储程序”是通用图灵机的基本想法，只是当时搞计算机的电子工程专家们没领会。von Nuemann基于其数学家的敏感，用工程师可以理解的语言把存储程序的想法重说了一遍。虽然von Neumann对现代电子计算机的发展功勋卓著，但他也很有自知之明。当有人要称他为“计算机之父”时，他立刻摆手说不敢当，并说这一荣誉“只能属于图灵”。

第三个问题是函数式编程。在常规语言里确实可以做一点函数式编程，但能做的事情很有限。以蔡先生提出的“无副作用”、“第一级函数”为例，拿C语言作为代表。对简单的基本类型，我们可以用函数式编程方式写出任何计算函数（理论保证），因为C语言为这些类型提供了许多基本运算，还要归功于C里的条件表达式和允许多个return出口（如果这两样都没有，不用赋值能写的东西就不值一提了）。但另一方面，在C里（和其他非函数式语言里）都缺乏操作复合数据的手段（除非通过赋值）。即使想写一个以结构（struct）为参数返回结构的函数，要将其中某成分加一，用函数式编程也很麻烦。对数组更是毫无办法，因为C数组不是一级对象，不用赋值什么事情也没法做（在其他语言里也没办法）。所谓“第一级函数”（简单点，“一级函数”）是指把函数也作为语言里的“一等公民”，能赋值，能在函数间传来传去，能生成新的“值”（这里也就是生成新函数了）。是否一等公民是语言的规定，如果一个语言的宪法（语言文本）规定函数不是一等公民，我们是没办法在该语言里把它当一等公民用的。例如在C语言里可以把函数指针当作函数的代表（多数主流语言不允许这一概念），但我们没办法让程序在运行中生成新函数。在没有上述C特征（条件表达式、函数的多出口、函数指针）的命令式语言里（Fortran、BASIC、Pascal、Ada 83等等），不用赋值能做的事就很可怜了。

另外关于函数式与命令式语言和栈与堆的关系，蔡先生说的也很不确切。历史（和现实）中存在许多没有堆的命令式语言（Fortran 90之前的Fortran，BASIC，Algol 60等等），但对于函数式语言，堆和栈“一个都不能少”。堆对函数式语言的重要性绝对是第一位的，废料收集（垃圾回收）就是为了堆管理（List的发明），可以说没有堆就没有函数式语言。说函数式语言中的数据主要放在栈中显然是不对的。函数式语言的所有数据结构都存在堆中，且完全自动分配和释放，无须人为干预。栈里（和命令式语言一样）存放的是函数的局部信息和控制信息。文中说函数式语言里是“由栈指向堆”就更难让人理解了。命令式语言里的堆主要也是由栈指向的，堆对象必须通过命名对象引用，在命令式语言或函数式语言里都一样。另外，函数式语言里一般都要有一个全局性的A表（可理解为原子表），程序运行中它引用着大量非临时性的堆数据。也就是说，函数式语言里可能存在大量不是由栈引用的堆对象。一个“全局”表达式求值完成，栈将变为空，但堆中还有大量活对象。这是技术细节，这里不仔细讨论了。■