《集异璧之大成》和编程递归

一、 集异璧和哥德尔不完备定理

GEB 这本书通过对于哥德尔的数理逻辑、艾舍尔的绘画、巴赫的音乐三者进行综合性的描述，引入了很多天马行空的故事，巧妙地介绍了很多关于数理逻辑学、可计算理论、人工智能、语言、音乐、绘画等方面的理论。作者通过巧妙地故事构思、广阔的视角和富于哲理的话题，让我们能通俗的理解很多计算机数理逻辑方面的知识。

这本书介绍了很多关于巴赫和埃舍尔的研究领域。他们相差三百多年，然而的研究领域中都包含了很多各种相似的结构—“无穷上升”。无穷上升有一个很有意思的特点，不停地自我复制，这样的自我复制和自我涉指，在数学中引起了巨大的革命，从 20 世纪初开始，大卫希尔伯特，康托，冯诺依曼等都研究过和递归相关的概念，直到最后哥德尔提出不完备定理，才让我们看到一个更大的世界。哥德尔的定理描述了一个符号系统，通常包括一组符号、一组公式和一组推导规则，就可以让我们用来推导定理了，这个定理发现对于这样一个复杂的符号系统，任意一个包含一阶谓词逻辑与初等数论的形式系统，都存在一个命题，它在这个系统中既不能被证明也不能被否定，除了第一不完备定理外，哥德尔第二不完备定理描述了如果系统 S 含有初等数论，当 S 无矛盾时，它的无矛盾性不可能在 S 内证明。这两个不完备定理描述了在一个不完备的系统中，这个系统无法解释自身的合理性。就像很久以前的说谎者悖论、理发师问题一样。这在编程中的递归有很重要的作用。

举很常识的例子，小明把苍老师的作品藏在了"d 盘-主题教育-核心价值观"文件内，那么在"核心价值观"文件夹内，小明可以对这些作品进行修改名字啦、移动顺序啦、点开看啦或者是删除等等操作，但小明在文件夹内，不能对这个文件夹本身进行操作，比如他不能在"核心价值观"文件夹内直接把"核心价值观"文件夹删掉。小明要想删除它，就必须到"主题教育"这一更高的层面去操作。这种性质叫做不完备性，这个名字是针对完备性（*）而言的。而哥德尔不完备性定理主要就在说：”大自然中的不完备性还真是无法抗拒、随处可见啊（**）。“换句更数学的话来说，这个定理允许了这样一个事实的存在：总有一些命题，在某一场合下，无法判断其真假性，即有些命题在某一场合下，你即可以说它是真命题，也可以说它是假命题，我们不能对其真伪性进行判断，它的真伪性不会对整个逻辑系统带来影响，但在更高的层面，命题说或许就能做出判断了。说句题外话，LZ 有没有考虑过这样的问题：什么是真理？真理都是可以被证明的吗？可以被证明的，就是所谓的真理吗？这里的证明就是指某种逻辑产物。现在哥德尔定理就说，（某一固定场合下）真理不都是可以被证明的（因为总有那么一些命题，我们无法得知其真伪性）；那反过来呢？TBBT 有集 Leonard 对 Penny 说，弦论（string theory）虽然现在内部逻辑系统自洽，但还不能说明它就是真的。所以能够被证明的，也未必都是真理。总之，哥德尔定理告诉了我们数学和逻辑的极限，这也几乎是人类理性的极限。它证明理性不是无所不能的，很多理性的系统是不完备的，需要引入人类的经验积累、实验模拟等方法来获得更多的认识。

二、 编程中的递归和递归论

在数理逻辑和可计算性中，递归函数是一类从自然数到自然数的函数，被证明是可计算的，递归函数接受有限元祖并返回单一自然数的偏函数。在编程中，程序调用自身的编程技巧称为递归。递归做为一种算法在程序设计语言中广泛应用。一个过程或函数在其定义或说明中有直接或间接调用自身的一种方法，它通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。一般来说，递归需要有边界条件、递归前进段和递归返回段。当边界条件不满足时，递归前进；当边界条件满足时，递归返回。

递归在数理逻辑中有着更为严格的证明，利用集合论等数学知识对于递归函数做了严格的定义，而在实际的编程应用中，递归函数对于解决实际问题也有非常好的效果，对于我们处理实际问题，将很多实际问题转化为递归问题来解决，可以帮助我们消除迭代方法中的过于繁琐问题，在实际的编程语言中，如 C、Java、Python 等语言都有各自的递归方案，对于解决实际的斐波那契数列、兔子繁殖、以及和很多组合数学中相关的递归公式等问题时，都有非常重要的作用，比如 Fib（n）=Fib（n-1）+Fib(n-2)就能很好的解决很多递归中的问题，可以让我们避免采用迭代算法中复杂的 for、while 等循环结构。

除了编程语言中经常实现的普通递归外，有一种非常重要的递归是尾递归，尾递归在很多编程语言中都有实现，尾递归比普通的递归有着更好的效果，因为尾递归能将一个递归的结果直接当作参数分配给下一个函数，避免了递归函数不断调用过程中对于计算机内存的消耗，大大减少了内存的消耗能让我们在使用计算机解决实际问题过程中获得更好的解决思路。在下图中，我们可以看到一个普通的递归函数以及这个函数在运行过程中计算机内存的变化，而更为聪明的计算机设计师不仅看到了递归对于一个内存栈的消耗，同时设计了新的递归函数来解决这个问题从而产生尾递归。尾递归能有效帮助递归函数减少对于内存的消耗。

```
def recsum(x):
  if x == 1:
    return x
  else:
    return x + recsum(x - 1)
```

```
recsum(5)
5 + recsum(4)
5 + (4 + recsum(3))
5 + (4 + (3 + recsum(2)))
5 + (4 + (3 + (2 + recsum(1))))
5 + (4 + (3 + (2 + 1)))
5 + (4 + (3 + 3))
5 + (4 + 6)
5 + 10
15
```

```
def tailrecsum(x, running_total=0):
  if x == 0:
    return running_total
  else:
    return tailrecsum(x - 1, running_total + x)
```

```
tailrecsum(5, 0)
tailrecsum(4, 5)
tailrecsum(3, 9)
tailrecsum(2, 12)
tailrecsum(1, 14)
tailrecsum(0, 15)
15
```

上图左边表示采用递归方法计算 1 到 5 的和，左边并未采用尾递归，导致每次在讲计算规模缩小的时候，必须要保存之前的数据结果，从而导致了内存的浪费，而采用尾递归后，模型发生了变化，不仅在内存上减少了消耗，而且在函数结构上也发生了变化，将原来保存在内存的结果当作参数传递给下一个递归函数，从而减轻了递归的调用，这样自然有利于结果的获取。在 GEB 这本书中，作者不仅举了很多设计递归函数的例子，同时也将计算机科学中的递归和音乐中的无穷阶升、绘画中的递归做了很多有趣的结合，这让我们认识到，世界上有很多相似的东西，这些相似的东西是我们学习计算机科学的思想源头，我们在利用计算机进行建模计算过程中，必须要能够结合实际生活中的很多实例，将这些实例转化为计算机模型，这样才能更好的解决问题。

三、 Lambda 演算和编程语言中的实现

除了作者研究的很多无穷递归之外，我也很感兴趣 lambda 演算。lambda 演算是一个形式系统（形式系统主要是由形式语言加上推理规则或转换规则构成的集合），它主要是被用来研究函数定义，函数应用和递归。简单点说，lambda 演算就是一个小的形式系统，它主要表达了计算机计算中两个概念：“代入”和“置换”。“代入”通俗点解释就是和我们平常接触的函数调用类似，比如用实参代入到形参。“置换”一般理解为变量换名规则。“代入”就是后面要讲到的 lambda 演算中的 β-规约，而“置换”相当于 lambda 演算中的 α--变换。

这一个 Lambda 演算在实际问题中有很重要的价值，随着计算机科学的发展，尤其是软件工程等在实践中发挥越来越大的作用，人们发现计算机的主要作用就是实现函数式编程，因此，在当前有很多人信奉函数是编程中的一等公民，各种编程语言中都有研究 Lambda 表达式的方式，采用 lambda 函数来实现变成成为当前很多面向对象函数的新方向。比如在最新的 java8 中，就借鉴了很多其他编程语言如 Lisp 等 Lambda 表达式。这样的表达式不仅能在数学上获得严格的证明，同时，在解决实际的编程问题中时，可以利用这种 lambda 表达公式的简洁性解决 OOP 语言中代码冗长的毛病。

比如在以往采用循环编写代码时，需要编写如下的代码实现。List<String> list = getMyStrings();for(String myString: list)……而采用 lambda 表达式后，代码可以进一步化简为如下特点，这样能充分利用函数式编程的好处解决实际问题。Predicate<String> matched = s -> s.equalsIgnoreCase(possible);在当前热门的大数据计算框架中，Spark 和 Hadoop 等技术为了能更好的适合分布式编程框架，需要借鉴函数式的编程逻辑，从而将 Lambda 表达式引入到这样的计算框架中，获得更好的编程性能。

四、 总结

GEB 这本书给了我很多不同的启发和思路。除了上面提到的两点，我们还能了解到很多关于人工智能的思想和思路。然而需要指出的是，当前随着计算机科学的发展，在人工智能方面，纯粹的依赖数理逻辑建立的知识结构或者图灵机等希望表达和建立的系统很难获得成功。如果想要获得人的思路和思想，仅仅设计一种模拟人思考的系统是不够的，我们需要让机器和人借鉴人类历史中的各种遗留的知识和思想，通过统计学习，从现有的知

识中获取到很多形式系统难以学习到的知识，只有这样，对于一个不完备的系统，才能变得更加像人工智能。人类采用自己独有的智慧才能解决诸多数理逻辑上未解决的问题。当前的世界随着移动互联网的发展，人们的数据量被史无前例的记录下来，而且在这个数据化浪潮中，数据中可以学习到很多通过形式语言和自动机难以获得的知识，在这样一个背景下，我们可以保存自己通过数理逻辑和可计算性中获得的理论，结合编程语言中的实际应用，将人工智能中的尾递归、递归、Lambda 表达式等工具应用到对于当前大数据背景中的实践中来，通过不断的实践应用，结合统计学知识，获得更大的成就。

除了我个人对于人工智能方面的思考之外，我们也要看到人类的创造性，在巴赫和埃舍尔等人的艺术成就中获得自己的精神慰藉。这些艺术学家利用自己对于生活中的事物观察和研究，创造出属于自己特色的艺术作品，从这一点上来说，艺术创造和科技创新是相同的，我们在科研中也要观察实际生活，利用生活中的例子解决计算机科学中的问题。当前，人工智能、机器学习等算法和方法获得了非凡的成就，我们要利用自己学到的数理逻辑基本原理，结合现实中的实际场景，在未来的科技世界占据一席之地。

本文参考资料或者文献：

https://www.zhihu.com/question/20569424

https://en.wikipedia.org/wiki/G%C3%B6del,_Escher,_Bach

https://en.wikipedia.org/wiki/Lambda_calculus

http://blog.csdn.net/neutblue/article/details/6848369