图灵机与计算问题

张江(email: jakezj@163.com)

自从 20 世纪 30 年代以来，图灵机、计算这些重要的概念在科学的天空中就一直闪烁着无限的光彩。尤其是近年来量子计算机、生物计算机、DNA 计算等领域的创新工作引起了世人的广泛关注。我们不禁问这样的问题，国外究竟为什么能发明出这些各式各样的计算机呢？这些意味着什么呢？其实这一切的源头都来源于计算理论。国内在介绍计算理论方面的教材虽然有不少，但一般都比较深奥难懂。所以我觉得很有必要对这些内容进行科普。于是尝试写下这么一篇文章，希望我的文章能够让你更加清楚、透彻的理解图灵机、计算等等一些基本而重要的概念，并洞悉到这些概念的本质和深远涵义。

本篇文章大体上可以分成四部分：首先给大家讲一讲关于图灵、哥德尔等科学家的故事；然后正式引入图灵机的概念，为了对这个概念有比较直观的理解，我采用了一个人工生命：“小虫”的比喻来叙述。接下来，文章介绍了跟图灵机有关的概念：什么是模拟，什么是“万能计算机”等等；最后是关于图灵停机问题的探讨，我个人认为很有可能未来对科学的重大突破都来源于对图灵停机问题的深入理解。在行文过程中，我除了用自己的方式介绍一些现有的基本概念之外（为了尽量表达得清楚明白，我不得不放弃理论论证的严格性），还探讨了很多我认为别人没有探讨的问题，这些问题多是我自己的思考结果，而它们没有经过科学的验证。在这部分内容上我都标上了*号，希望你能有选择的看待这些问题和观点。

一、故事

任何科学思想、科学概念的诞生都有它的背景，在背景中往往有很多迷人的故事。关于计算理论可以追溯到 1900 年，当时著名的大数学家希尔伯特在世纪之交的数学家大会上给国际数学界提出了著名的 23 个数学问题。其中第十问题是这样的：存在不存在一种有限的、机械的步骤能够判断“丢番图方程”是否存在解？这里就提出来了有限的、机械的证明步骤的问题，用今天的话说就是算法。但在当时，人们还不知道“算法”是什么。实际上，当时数学领域中已经有很多问题都是跟“算法”密切相关的，因而，科学的 “算法” 定义呼之欲出。之后到了 30 年代的时候，终于有两个人分别提出了精确定义算法的方法，一个人是图灵，一个人是丘奇。而其中图灵提出来的图灵机模型直观形象，于是很快得到了大家的普遍接受。

不知道你是否听说过图灵这个名字。可能有些人知道牛顿，知道爱因斯坦，甚至知道冯诺依曼，但不知道图灵。然而图灵的贡献绝对不亚于这些科学大师。图灵最大的贡献就是把算法这样一个基本的、深刻的概念用他的图灵机模型讲清楚了。正是因为图灵奠定的理论基础，人们才有可能发明 20 世纪以来甚至是人类有史以来最伟大的发明：计算机。因此人们称图灵为：计算机理论之父。

图灵生活的年代经历了第二次世界大战。在二战期间他曾经为英国政府效力成功破译了德国的密码，因而为英国做出了突出贡献。其实也正是因为二战，英国政府才肯掏钱让图灵制造最原始的计算机，当然这种计算机是专门用来破译密码用的，而不是我们现在用的通用计算机。（有一部片子叫《密码迷情》英文名是《enigma》就是根据图灵当时破译德国密码的故事改编的，大家有兴趣可以去找一找。）

图灵这个人很古怪，只喜欢自己一个人闷头研究，不喜欢与别人交流。并且据说他还是一个同性恋者。要知道在当时的英国，同性恋行为可是大逆不道的。最后，在他事业刚刚达到顶风的时候，他自杀了。为了纪念这个伟大的学者，计算机界设立了最高荣誉奖：ACM 图灵奖。

图灵机的产生一方面奠定了现代数字计算机的基础(要知道后来冯诺依曼就是根据图灵的设想才设计出第一台计算机的)。另一方面，根据图灵机这一基本简洁的概念，我们还可以看到可计算的极限是什么。也就是说实际上计算机的本领从原则上讲是有限制的。请注意，这里说到计算机的极限并不是说它不能吃饭、扫地等硬件方面的极限，而是仅仅就从信息处理这个角度，计算机也仍然存在着极限。这就是图灵机的停机问题。这个问题在图灵看来更加重要，在他当年的论文中，其实他是为了论证图灵停机问题才“捎带手”提出了图灵机模型的。

提到了图灵停机问题，我不禁又要提一提哥德尔定理、罗素悖论、康托尔的集合论等等一系列大事儿。早在 19 世纪末的时候，康托尔为集合论做了奠基性的研究。要知道，数学虽然五花八门，但是人们发现，运用集合这个概念可以概括所有的数学，也就是说集合是一切数学的基础。因而如果为集合论奠定了公理化的基础，也就等于为数学奠定了基础。康托尔就是做了这方面的贡献。另外，他为了证明实数的个数比自然数多这个结论，发明了一种被称为“对角线删除”的证明方法。没想到的是，这个方法影响非常深广，直到后来的图灵停机问题、哥德尔定理其实都是该方法的不同延伸。

19 世纪末的人们忙于为基于集合论的数学建立公理体系大厦。然而就当这座大厦即将完工的时候，一件可怕的事情发生了，罗素提出来的罗素悖论粉碎了数学家的梦想。关于罗素悖论的一个通俗化版本是：“村子里有一个理发师，他给自己定了一条规矩：‘不给那些所有给自己理发的人理发’。现在就要问，这个理发师该不该给自己理发？”。如果你尝试回答这个问题就会发现奇怪的事情：这个问题本身似乎是不可能的！正是因为这种奇怪的逻辑，哲学家罗素才颠覆了整个数学大厦的基础！

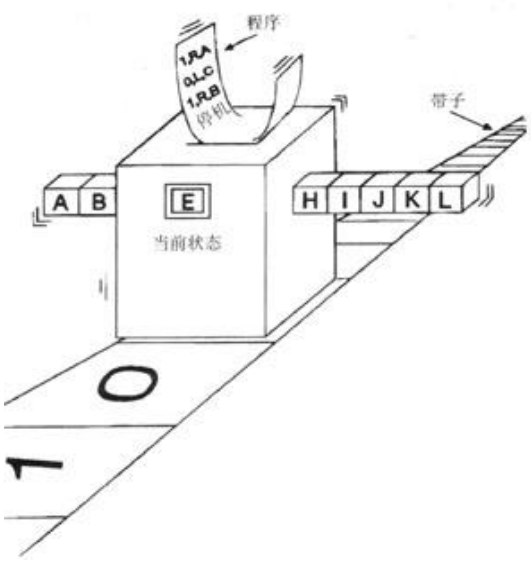
因为集合论中存在着矛盾，所以整个数学体系存在着根本性的矛盾，然而具有讽刺意味的是，数学一直以严格著称！这种感觉就好像正当你得意洋洋的时候有人突然闪了你一个耳光！人们在一阵慌乱之后开始逐渐稳下阵角寻找避免罗素悖论产生的方法，并总希望通过细心的选择数学公理能够将类似罗素悖论这样的怪物一劳永逸的排除精确数学体系之外。这就是后来希尔伯特提出来一套数学纲领的原因，他希望找到一套公理体系能够排除悖论，并挽救纯粹而美丽无暇的数学。虽然希尔伯特没能完成他的梦想，但他坚信梦想是对的。

然而，没过多少年，一个名不见经传的年轻逻辑学家哥德尔提出来的定理却彻底粉碎了希尔伯特的梦。这就是后来著名的哥德尔定理！该定理大致上说：任何一个数学的公理化体系都不是“完美的”。换个角度说，任何数学公理化系统都是死的，它总需要人为地从外界注入新的公理进去才能让它日趋完善，而它自己并不能完全自动避免矛盾产生。哥德尔定理可以说整个扭转了人们的世界观。因为被认为最“完美”最“纯粹”的数学都是不完全的，那么纯粹完美的世界也应该不存在。哥德尔定理还指出了“理性”和“分析”方法的极限。这才是后来人们步入到了“综合”时代的部分原因。更有趣的是，哥德尔用来证明他定理的方法正和康托尔证明实数比自然数多、图灵停机问题以及罗素悖论的方法是一脉相承的。

所有这些都是 20 世纪上半个世纪发生的大事，后来发生了什么？计算机出现了、信息时代来临了，似乎科学技术是万能的，它们总会改善我们的生活，满足我们的欲望。渐渐的，人们似乎淡忘了图灵、歌德尔、康托尔等等大师们的思想了。然而近年来随着复杂性科学的研究，人们却又有了重拾这些相对古老而根本问题的迹象了！

二、图灵机

下面言归正传，我们开始讲图灵机的概念。我先把图灵机的模型给你，虽然有些无趣，不过请坚持看下去，我会在下面运用大家比较好理解的形式重新解释的。在这里你仅仅需要认识它的轮廓。一个图灵机是形如下面的一个装置：



这个装置由下面几个部分组成：一个无限长的纸带，一个读写头。（中间那个大盒子），内部状态（盒子上的方块，比如 A,B,E,H），另外，还有一个程序对这个盒子进行控制。这个装置就是根据程序的命令以及它的内部状态进行磁带的读写、移动。

它工作的时候是这样的：从读写头在纸带上读出一个方格的信息，并且根据它当前的内部状态开始对程序进行查表，然后得出一个输出动作，也就是是否往纸带上写信息，还是移动读写头到下一个方格。程序也会告诉它下一时刻内部状态转移到哪一个。

具体的程序就是一个列表，也叫做规则表，是这样的：

当前内部状态 s	输入数值 i	输出动作 o	下一时刻的内部状态 s'
B	1	前移	C
A	0	往纸带上写 1	B
C	0	后移	A
…	…	…	…

因此，图灵机只要根据每一时刻读写头读到的信息和当前的内部状态进行查表就可以确定它下一时刻的内部状态和输出动作了。

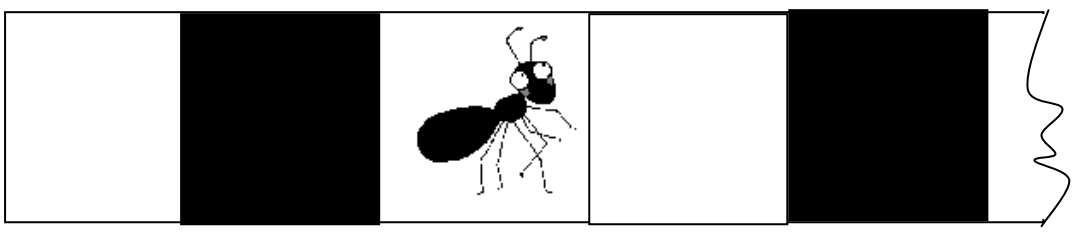
图灵机就是这么简单！不可思议吧？而只要你变化它的程序（也就是上面的规则表），那么它就可能为你做任何计算机能够完成的工作。因此可以说，图灵机就是一个最简单的计算机模型！

也许，你会觉得图灵机模型太简单，怎么可能完成计算机的复杂任务呢？问题的关键是如何理解这个模型。

三、如何理解图灵机？

1、小虫的比喻

我们不妨考虑这样一个问题。假设一个小虫在地上爬，那么我们应该怎样从小虫信息处理的角度来建立它的模型？



首先，我们需要对小虫所在的环境进行建模。我们不妨就假设小虫所处的世界是一个无限长的纸带，这个纸带上被分成了若干小的方格，而每个方格都仅仅只有黑和白两种颜色。很显然，这个小虫要有眼睛或者鼻子或者耳朵等等感觉器官来获得世界的信息，我们不妨把模型简化，假设它仅仅具有一个感觉器官：眼睛，而且它的视力短得可怜，也就是说它仅仅能够感受到它所处的方格的颜色。因而这个方格所在的位置的黑色或者白色的信息就是小虫的输入信息。

另外，我们当然还需要为小虫建立输出装置，也就是说它能够动起来。我们仍然考虑最简单的情况：小虫的输出动作就是往纸带上前爬一个方格或者后退一个方格。

仅仅有了输入装置以及输出装置，小虫还不能动起来，原因很简单，它并不知道该怎样在各种情况下选择它的输出动作。于是我们就需要给它指定行动的规则，这就是程序！假设我们记小虫的输入信息集合为 I={黑色，白色}，它的输出可能行动的集合就是：O={前移，后移}，那么程序就是要告诉它在给定了输入比如黑色情况下，它应该选择什么输出。因而，一个程序就是一个从 I 集合到 O 集合的映射。我们也可以用列表的方式来表示程序，比如：

程序 1：

输入	输出
黑色	前移
白色	后移

这个程序非常简单，它告诉小虫当读到一个黑色方格的时候就往前走一个方格，当读到一个白色方格的时候就后退一个格。

我们不妨假设，小虫所处的世界的一个片断是：黑黑黑白白黑白……，小虫从左端开始。

那么小虫读到这个片断会怎样行动呢？它先读到黑色，然后根据程序前移一个方格，于是就会得到另外一个黑色信息，这个时候它会根据程序再次前移一个方格，仍然是黑色，再前移，这个时候就读到白色方格了，根据程序它应该后退一个格，这个时候输入就是黑色了，前移，白色，后移……，可以预见小虫会无限的循环下去。

然而，现实世界中的小虫肯定不会这样傻的在那里无限循环下去。我们还需要改进这个最简单的模型。首先，我们知道小虫除了可以机械地在世界上移动以外，还会对世界本身造成影响，因而改变这个世界。比如虫子看到旁边有食物，它就会把那个东西吃掉了。在我们这个模型中，也就相当于我们必须假设小虫可以改写纸带上的信息。因而，小虫可能的输出动作集合就变成了：O={前移，后移，涂黑，涂白}。这个时候，我们可以把程序 1 改为比如：

程序 2：

输入　输出

黑　　前移

白　　涂黑

纸带是黑黑白白黑……，小虫会怎样行动呢？下面的图表示出了这个例子中每一步小虫的位置（标有圆点的方格就是小虫的当前位置），以及纸带的状况。

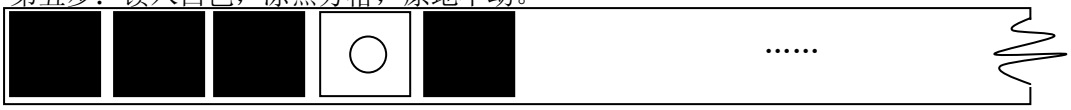
开始：小虫在最左边的方格，根据程序的第一行，读入黑色应该前移。

第二步：仍然读入黑，根据程序的第一行，前移。

第三步：这个时候读入的是白色，根据程序的第二行，应该把这个方格涂黑，而没有其他的动作。假设这张图上方格仍然没有涂黑，而在下一时刻才把它表示出来。

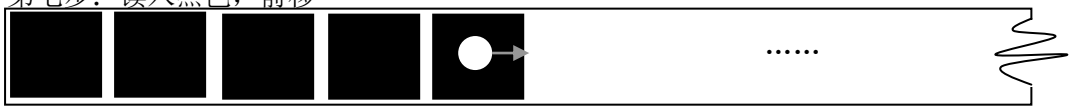
第四步：当前方格已经是黑色的，因此小虫读入黑色方格，前移。

第五步：读入白色，涂黑方格，原地不动。



第六步：当前的方格已经被涂黑，继续前移。

第七步：读入黑色，前移



小虫的动作还会持续下去……。我们看到，小虫将会不停的重复上面的动作不断往前走，并会把所有的纸带涂黑。

显然，你还可以设计出其他的程序来，然而无论你的程序怎么复杂，也无论纸带子的情况如何，小虫的行为都会要么停留在一个方格上，要么朝一个方向永远运动下去，或者就是在几个方格上来回打转。然而，无论怎样，小虫比起真实世界中的虫子来说，还有一个致命的弱点：那就是如果你给它固定的输入信息，它都会给你固定的输出信息！因为我们知道程序是固死的，因此，每当黑色信息输入的时候，无论如何它都仅仅前移一个方格，而不会做

出其他的反应。它似乎真的是机械的！

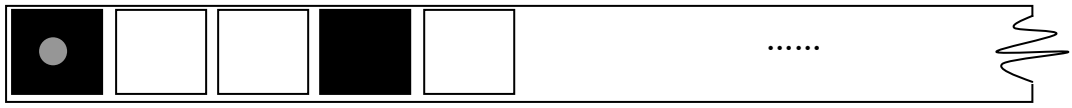
如果我们进一步更改小虫模型，那么它就会有所改进，至少在给定相同输入的情况下，小虫会有不同的输出情况。这就是加入小虫的内部状态！我们可以作这样的一个比喻：假设黑色方格是食物，虫子可以吃掉它，而当吃到一个食物后，小虫子就会感觉到饱了。当读入的信息是白色方格的时候，虽然没有食物但它仍然吃饱了，只有当再次读入黑色时候它才会感觉到自己饥饿了。因而，我们说小虫具有两个内部状态，并把它内部状态的集合记为：S={饥饿，吃饱}。这样小虫行为的时候就会不仅根据它的输入信息，而且也会根据它当前的内部状态来决定它的输出动作，并且还要更改它的内部状态。而它的这一行动仍然要用程序控制，只不过跟上面的程序比起来，现在的程序就更复杂一些了，比如：

程序 3：

输入	当前内部状态	输出	下时刻的内部状态
黑	饥饿	涂白	吃饱
黑	吃饱	后移	饥饿
白	饥饿	涂黑	饥饿
白	吃饱	前移	吃饱

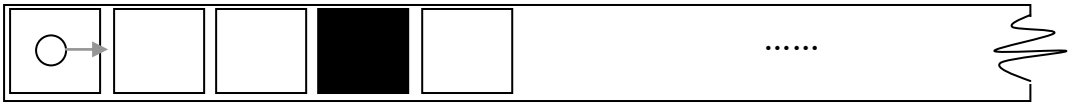
这个程序复杂多了，有四行，原因是你不仅需要指定每一种输入情况下小虫应该采取的动作，而且还要指定在每种输入和内部状态的组合情况下小虫应该怎样行动。看看我们的虫子在读入黑白白黑白……这样的纸带的时候，会怎样？仍然用下面的一系列图来表示，灰色的圆点表示饥饿的小虫，白色的圆点表示它吃饱了。为了清晰，我们把小虫将要变成的状态写到了图的下一行。

假定它仍然从左端开始，而且开始的时候小虫处于饥饿状态。这样读入黑色，当前饥饿状态，根据程序第一行，把方格涂白，并变成吃饱（这相当于把那个食物吃了，注意吃完后，小虫并没动）。



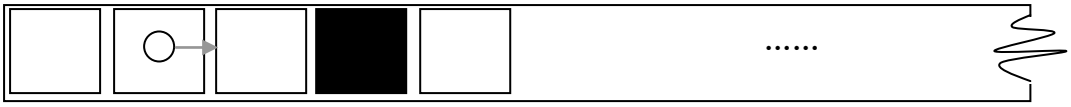
涂白方格，变成吃饱

第二步：当前的方格变成了白色，因而读入白色，而当前的状态是吃饱状态，那么根据程序中的第四条前移，仍然是吃饱状态；



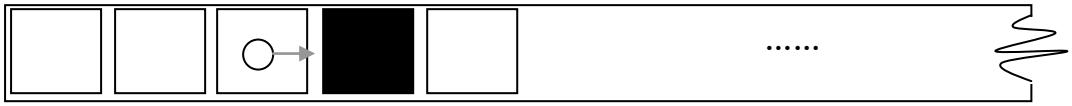
前移，仍然吃饱

第三步：读入白色，当前状态是吃饱，因而会重复第二步的动作。



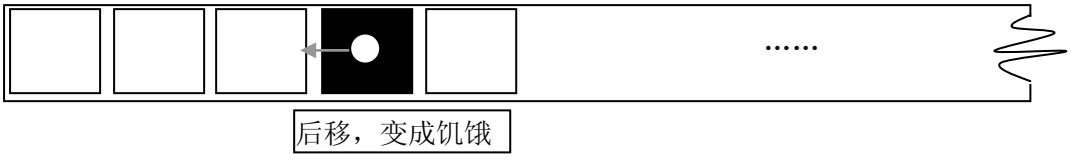
前移，保持吃饱

第四步：仍然重复上次的动作。

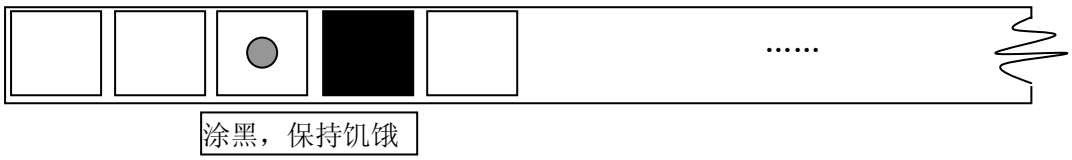


前移，保持吃饱

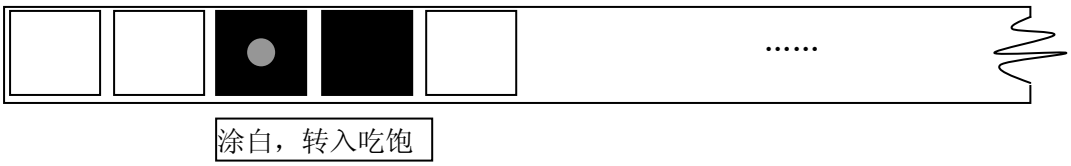
第五步：读入黑色，当前状态是吃饱，这时候根据程序的第二行应该后移方格，并转入饥饿状态；



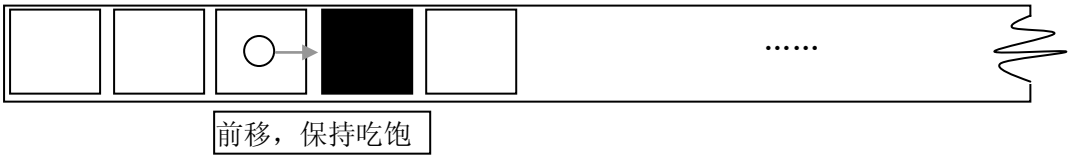
第六步：读入白色，当前饥饿状态，根据程序第三行应该涂黑，并保持饥饿状态（各位注意，这位小虫似乎自己吐出了食物！）；



第七步，读入黑色，当前饥饿，于是把方格涂白，并转入吃饱状态（呵呵，小虫把刚刚自己吐出来的东西又吃掉了！）。



第八步，读入白色，当前吃饱，于是前移，保持吃保状态。



这时候跟第四步的情况完全一样了，因而小虫会完全重复 5、6、7、8 步的动作，并永远循环下去。似乎最后的黑色方格是一个门槛，小虫无论如何也跨越不过去了。

小虫的行为比以前的程序复杂了一些。尽管从长期来看，它最后仍然会落入机械的循环或者无休止的重复。然而这从本质上已经与前面的程序完全不同了，因为当你输入给小虫白色信息的时候，它的反应是你不能预测的！它有可能涂黑方格也有可能前移一个。当然前提是你不能打开小虫看到它的内部结构，也不能知道它的程序，那么你所看到的就是一个不能预测的满地乱爬的小虫。如果小虫的内部状态数再增多呢，那么它的行为会更加的不可预测！

说到这里，你可能对于“小虫的行为不可预测”这句话持反对意见。因为所有可能的输入状态是固定的，所有的内部状态无论多少也是固定的，那么小虫所有可能的行为就应该是有限的！然而，不要忘记纸带的长度是无限的，因而虽然每个具体的输入可能就有 0 和 1 两种状态，然而这些 0 和 1 的输入组合却是无限的。另外退一步说，输入纸带的情况是有限的（你可以理解为 01 组合经过若干长度就会出现循环，比如 011011011…），那么我们的小虫会不会必然最终陷入到无休止的循环中呢？答案是肯定的，因为这个时候输入的组合数乘以内部状态总数是一个有限的数值，因而小虫必然会遍历所有的可能情况。无论哪种情况下，你有可能都会认为小虫是可以预测的，也就是似乎你可以通过某种聪明的“数学”判断小虫是否会循环，以及在什么时候循环。也就是说，通过你的聪明的数学，你只要看看小虫的程序，而不用执行它就能够预言小虫在多少多少步之后必然会“傻傻地”重复以前的动作了。这样一来，它可真是名副其实的“雕虫小技”了。然而真的是这样么？这种判定小虫傻循环的一般定理或程序存在么？这个问题留待我们后面进行讨论。

好了，如果你已经彻底搞懂了我们的小虫是怎么工作的，那么你已经明白了图灵机的工作原理了！因为从本质上讲，最后的小虫模型就是一个图灵机！

2、如何理解图灵机模型*

刚才用小虫说明了图灵机的工作原理，相信你的第一个反映就是，这样的模型太简单了！他根本说明不了现实世界中的任何问题！下面，我就要试图说服你，图灵机这个模型是伟大的！

首先，我想说的是，其实我们每一个会决策、会思考的人就可以被抽象的看成一个图灵机。

为什么可以做这种抽象呢？首先我们可以考虑扩展刚才说的小虫模型。因为小虫模型是以一切都简化的前提开始的，所以它的确是太太简单了。然而，我们可以把小虫的输入集合、输出行动集合、内部状态集合进行扩大，这个模型就一下子实用多了。首先，小虫完全可以处于一个三维的空间中而不是简简单单的纸带。并且小虫的视力很好，它一下子能读到方圆500 米的信息，当然，小虫也可以拥有其他的感觉器官，比如嗅觉、听觉等等，而这些改变都仅仅是扩大了输入集合的维数和范围，并没有其他更本质的改变。同样道理，小虫可能的输出集合也是异常的丰富，它不仅仅能移动自己，还可以尽情的改造它所在的自然界。进一步的，小虫的内部状态可能非常的多，而且控制它行为的程序可能异常复杂，那么小虫会有什么本事呢？这就很难说了，因为随着小虫内部的状态数的增加，随着它所处环境的复杂度的增加，我们正在逐渐失去对小虫行为的预测能力。但是所有这些改变仍然没有逃出图灵机的模型：输入集合、输出集合、内部状态、固定的程序！就是这四样东西抓住了小虫信息处理的根本。

我们人能不能也被这样的抽象呢？显然，输入状态集合就是你所处的环境中能够看到、听到、闻到、感觉到的所有一起，可能的输出集合就是你的每一言每一行，以及你能够表达出来的所有表情动作。内部状态集合则要复杂得多。因为我们可以把任意一个神经细胞的状态组合看作是一个内部状态，那么所有可能的神经细胞的状态组合将是天文数字！

似乎你会说，这个模型根本不对，还有很多思维本质的东西没有概括进去。比如记忆问题，人有记忆，图灵机有么？其实，只要图灵机具有了内部状态，它就相应的具有了记忆。比如上面讲到的具有饥饿和吃饱两种状态的小虫就会记住它所经历过的世界：如果吃到食物就用吃饱状态来“记住”吃过的食物。什么是记忆呢？假如你经历了一件事情并记住了它，那么只要你下一次的行动在相同条件下和你记住这件事情之前的行动不一样了，就说明该事情对你造成了影响，也就说明你确实记住了它。

学习的问题反映在模型中了么？学习是怎么回事儿呢？似乎图灵机模型中不包括学习，因为学习就意味着对程序的改变，而图灵机是不能在运行过程中改变它的程序的。然而，我们不难假设，你实际上并不能打开一个人的脑袋来看，所以它的实际程序规则你是不知道的。很有可能一个图灵机的规则没有改变，只不过激活了它的某些内部状态，因而它的行为发生了本质上的变化，尽管给它相同的输入，它给出了完全不同的输出，因而在我们看来，它似乎会学习了！而实际上，这个图灵机的程序一点都没变。

还有很多很多现象似乎都能被图灵机包括，什么是人类的情绪、情感？你完全可以把它看作是某种内部状态，因而处于心情好的情绪下，你的输入输出是一套规则，而心情不好的时候则完全是另一套。这仍然没有逃出图灵机的模型范围。

接下来的问题就是我们人的思维究竟是不是和图灵机一样遵循固定的程序呢？这个问题似乎初看是不可能的，因为人的行为太不固定了！你不可预言它！然而我会争辩道，无论如何神经元传递信息、变化状态的规律都是固定的，可以被程序化的，那么作为神经元的整体：脑的运作必然也要遵循固定的规则也就是程序了。那么，如果是这样，正如图灵相信的，

人脑也不会超越图灵机这个模型，所以，人工智能也必然是可能的！然而，我认为针对这个问题的答案很有可能没有这么简单，我们将在最后详细讨论这个问题。

无论如何，我相信你已经能够体会到了，图灵机模型实际上是非常强有力的！

三、计算

1、什么是计算

说了这么多，虽然也许你已经了解到了图灵机的威力，也许还将信将疑，然而，你肯定仍然看不出来图灵机和计算有什么关系。而实际上，图灵机是一个理论计算机模型，它最主要的能耐还是在于计算上！所以，下面我们就来看看什么是计算！

我可以先给出一个很摩登的对计算概念的理解：广义上讲，一个函数变化如把 x 变成了 f(x)就是一个计算！如果我们把一切都看作是信息，那么更精确的讲，计算就是对信息的变换！如果采用这种观点，你会发现，其实自然界充满了计算！如果我们把一个小球扔到地上，小球又弹起来了，那么大地就完成了一次对小球的计算。因为你完全可以把小球的运动都抽象成信息，它无非是一些比如位置、速度、形状等等能用信息描述的东西嘛，而大地把小球弹起来就无非是对小球的这些信息进行了某种变换，因而大地就完成了一次计算！你可以把整个大地看作是一个系统，而扔下去的小球是对这个系统的输入，那么弹回来的小球就是该系统的输出，因而也可以说，计算就是某个系统完成了一次从输入到输出的变换！

这样理解不要紧，你会发现，现实世界到处都是计算了！因为我们完全可以把所有的自然界存在的过程都抽象成这样的输入输出系统，所有的大自然存在的变量都看作是信息，因而计算无处不在！也的确，正是采取了这样的观点，国外才有可能发明什么 DNA 计算机、生物计算机、量子计算机这些新鲜玩艺！因为人家把 DNA 的化学反应、量子世界的波函数变换都看作是计算了，自然就会人为地把这些计算组合起来构成计算机了。然而，似乎我们的理论家们还在力图证明关于图灵机的某个定理呢，却完全没有意识到计算其实就是这样简单！

下面回到图灵机！为什么说图灵机是一个计算的装置呢？很简单，图灵机也是一个会对输入信息进行变换给出输出信息的系统。比如前面说的小虫，纸带上的一个方格一个方格的颜色信息就是对小虫的输入，而小虫所采取的行动就是它的输出。不过这么看，你会发现，似乎小虫的输出太简单了。因为它仅仅就有那么几种简单的输出动作。然而，不要忘了，复杂性来源于组合！虽然每一次小虫的输出动作很简单，然而当把所有这些输出动作组合在一起，就有可能非常复杂！比如我们可以把初始时刻的纸带看作是输入信息，那么经过任意长的时间比如说 100 年后，小虫通过不断的涂抹纸带最后留下的信息就是输出信息了。那么小虫完成的过程就是一次计算。事实上，在图灵机的正规定义中，存在一个所谓的停机状态，当图灵机一到停机状态，我们就认为它计算完毕了，因而不用费劲的等上 100 年。

2、计算的组合

更有意思的是，我们可以把若干个计算系统进行合并构成更大的计算系统。比如还是那个小球吧，如果往地上放了一个跷跷板，这样小球掉到地上会弹起这个跷跷板的另一端，而跷跷板的另一边可能还是一个小球，于是这个弹起的小球又会砸向另一个跷跷板……。

我们自然可以通过组合若干图灵机完成更大更多的计算，如果把一个图灵机对纸带信息变换的结果又输入给另一台图灵机，然后再输入给别的图灵机……，这就是把计算进行了组合！也许你还在为前面说的无限多的内部状态，无限复杂的程序而苦恼，那么到现在，你不难明白，实际上我们并不需要写出无限复杂的程序列表，而仅仅将这些图灵机组合到一起就可以产生复杂的行为了。

有了图灵机的组合，我们就能够从最简单的图灵机开始构造复杂的图灵机。那么最简单的图灵机是什么呢？我们知道最简单的信息就是 0 和 1，而最简单的计算就是对 0 或 1 进行布尔运算。而布尔运算本质上其实就三种：与、或、非。从最简单的逻辑运算操作最简单的二进制信息出发我们其实可以构造任意的图灵机!这点不难理解：任何图灵机都可以把输入、输出信息进行 01 的编码，而任何一个变换也可以最终分解为对 01 编码的变换，而对 01 编码的所有计算都可分解成前面说的三种运算。也许，现在你明白了为什么研究计算机的人都要去研究基本的布尔电路。奥秘就在于，用布尔电路可以组合出任意的图灵机！

3、征服无限的方法！

回忆你小时候是如何学会加法运算的。刚开始的时候，你仅仅会死记硬背。比如你记住了 1+1=2，记住了 2+4=6，……。然而无论你记住多少固定数字的运算，你都不叫学会了加法。原因很简单，假如你记住了 n 对数的加法，那么我总会拿出第 n+1 对数是你没有记住的，因此你还是不会计算。原则上。自然数的个数是无穷的，所以任何两个数的加法可能结果也是无穷的，而如果采用死记硬背的方法，我们头脑怎么可能记住无穷数字的计算法则呢？但是随着年龄的增长，你毕竟还是最终学会了加法运算！说来奇怪，你肯定明白其实加法运算并不需要记住所有数字的运算结果，而仅仅需要记住 10 以内的任意两个数的和，并且懂得了进位法则就可以了。

你是怎么做到的呢？假设要计算 32+69 的加法结果，你会把 32 写到一行，把 69 写到下一行，然后把他们对齐。于是你开始计算 2+9=11，进一位，然后计算 3+6=9，再计算 9+1=10 再进一位，最后，再把计算的这些每一位的结果都拼起来就是最终的答案 101。这个简单例子给我们的启发就是：作加法的过程就是一个机械的计算过程，这里输入就是 32 和 69 这两个数字，输出就是 101。而你的程序规则就是具体的把任意两个 10 以内的数求和。这样，根据固定的加法运算程序你可以计算任两个数的加法了。

不知你发现了没有，这个计算加法的方法能够让你找到运用有限的规则应对无限可能情况的方法！我们刚才说了，实际上自然数是无限的，这样，所有可能的加法结果也是无限的。然而运用刚才说的运算方法，无论输入的数字是多少，只要你把要计算的数字写下来了，就一定能够计算出最终的结果，而无需死记硬背所有的加法！

因而，可以说计算这个简单的概念，是一种用有限来应对无限的方法！我们再看一个例子：假如给你一组数对：1,2　3,6　5,10　18,36，就这 4 对，这时问你 102 对应的数是多少？很显然，如果仅仅根据你掌握的已知数对的知识，是不可能知道答案的，因为你的知识库里面没有存放着 102 对应数字的知识。然而，如果你掌握了产生这组数对的程序法则，也就是看到如果第一个数是 x，那么第二个数就是 2x 的话，你肯定一下子就算出 102 对应的是 204 了。也就是说，你实际上运用 2x 这两个字符就记住了无限的诸如 1,2　3,6　102,204 所有这样的数对。

这看起来似乎很奇怪。我怎么可能运用有限的字符来应对无限种可能呢？实际上，当没有人问你问题的时候，你存储的 2x 什么也没有，而当我问你 102 对应的是多少？我就相当于给你输入了信息：102，而你仅仅是根据这个输入信息 102 进行一系列的加工变换得到了输出信息 204。因而输入信息就好比是原材料，而你的程序规则就是加工的方法，只有在原材料上进行加工，你才能输出最终产品。

这让我不禁想起了专家系统方法。其实专家系统就是一个大的规则库。也就相当于存储了很多很多的 1,2 3,6 5,10 这样特殊的规则对。而无论它存储的东西再多，总归会是有限的，你只要找到一个它没有存储到的问题，它就无能为力了。因而专家系统就会在你问到 102 对应是多少的时候失败！如何解决问题？人们想出了很多方法，就比如元规则的方法，其实元规则就相当于刚才所说的计算加法的程序，或者 2x 这样的东西。运用元规则的确可

以应对无限种情况了。所以，这就是为什么你问计算机任何两个数相加是多少，它总能给出你正确的答案的原因，虽然它不必记住所有这些加法对的信息。

然而仅仅是元规则就能解决所有问题么？

假如给你三组数对，排列成一个表：

1,2　3,6　4,8　　100,200

3,9　2,6　8,24　100,300

1,4　2,8　3,12　100,400

那么问你在第 6 行上，3 这个数字对应的是多少？我们先要找出第一行的规律是 2x 没有疑问，第二行呢？是 3x，第三行是 4x，那么第 6 行就应该是 7x 了，因而在第 6 行上 3 应该对应的是 21 了！这里跟前面不太一样的是，虽然我们得到了每一行的规则比如第一行的 2x，但是随着行数的增加，这个规则本身也变化了，因而第 2 行是 3x，第 3 行是 4x 等等，因而我们又得到了一个规则本身的规则，即如果行数是 n 的话，那么这一行的规则就是(n+1)x。我们显然能够根据输入的 n 和 x 计算出数值。把这个道理放到专家系统里面，这种原理就是元规则的规则，元规则的元规则……，应该是无穷的！然而专家系统本身并不会自动的归纳这些规则，人必须事先把这些元规则写到程序里，这也就是专家系统最大的弊端。而我们人似乎总能在一些个别的事件中归纳出规则。进一步问，机器可以归纳么？这就相当于说：可以为归纳方法编出程序么？这也是一个很有趣的问题，下面将要详细讨论！可以设想，假如我们找到了真正归纳的方法，那么编写出这样的程序，它就会一劳永逸的自己进行学习归纳了。我们完全再也不用给他编制程序和规则了。这正是人工智能的终极目标！

4、归纳*

记得金大侠在他的一本武侠小说：《倚天屠龙记》中曾讲述了这样一段故事：武林泰斗张三丰在情急之下要把他新创的武功“太极拳”传授给新起之秀张无忌。张无忌除了有一身精湛的“内功修为”以外还对武学具有极高的悟性。因而当张三丰给他打过一趟太极拳以后，他就把所有的招式全部记下来了并且当场把所学的太极拳重新再打给张三丰看。在张无忌练拳的过程中，张三丰反复问他一个问题：“你已经忘掉几招了？”。他的回答令其他人异常不解，因为他越在那里揣摩太极拳的奥秘，忘记的招数也越来越多。旁边的人不明白，这样的学法忘的这么快，怎么可能学会武功呢？然而，没过多长时间，张无忌说已经忘掉了所有的招式。张三丰笑着说：“不错，你终于学会了‘太极拳’”。

从这个例子中，我们看到了什么？张无忌之所以能学会太极拳，正是因为他已经能够从具体的一招一式之中抽象出了更高一层次的武学规律，因而，当他把所有的有形的武功招数都忘记的时候，已经掌握了太极拳的精髓。而太极武功讲究的就是借力打力，以柔克刚。说白了就是事先并没有固定招式存在，而等到敌人向我进攻的时候我再动态的生成破解的招术。

用到图灵机模型中，我们不难发现，如果把具体的武功招术比喻成一些输入，而应对招术比喻成图灵机的输出，那么太极所讲究的借力打力、以柔克刚的方法其实就是类似上节讲过的 2x 这样的图灵程序！因而张无忌学太极拳的过程就是从特殊的输入输出提升到了一般的算法的过程。也可以说，张无忌运用了归纳学习法！

然而，仔细观察上一节的叙述，我们会发现。虽然图灵机能够将 2x 这样的法则计算得出结果，但是抽象出 2x 本身并不是机器自动产生的，而是需要我们外在的人编程进去。那么，面对这样的问题，究竟图灵机能不能像张无忌一样进行归纳思维呢？

可以设想，如果计算机真有了张无忌那两下子，我们人类可要省事儿多了！我们甚至不需要为计算机编程序，它就会自动的从若干个具体事例中归纳出一般的通用规律来。然而，究竟计算机能不能具有真正的归纳能力呢？让我们来仔细考虑一下这个问题。

我们说如果计算机能自动归纳，也就意味着我们可以为归纳方法来编写一段程序 P。这个程序可以理解为输入的是一些特殊的数对，输出的是能够生成这些数对的程序。也就是说输入具体的“招术”，输出的是这些“招术”的一般规律。如果说程序 P 真正可以自己归纳，那么 P 就必然可以归纳出所有的规律。我们已经讨论过了，其实任何一个程序都能够被看作是对输入的一个变换而得到输出。那么程序 P 自然也是。假设这些对子(a,b),(c,d),(e,f),……都是程序 P 的输入输出对，那么我们挑选出前 1000 个（总而言之是足够多的对子）。把这 1000 个特殊情况输入到 P 中，那么 P 就应该能够产生这些对子的共性，也就是 P 自己这个程序了！换句话说，程序 P 产生了它自己，P 自己把自己给归纳出来了！这似乎陷入了怪圈之中！另外，我们人类设计出来 P，如果 P 可以归纳所有的规律，那么 P 能否也能归纳出“人归纳 P”本身这个规律呢？仍然是怪圈问题！这样的问题似乎还有很多。反过来讲，如果假设归纳出所有规律的程序 P 不存在，那么为什么我们人类总能归纳出规律呢？什么样的具体问题是可归纳的，什么问题是不可归纳的？然而这些看起来非常重要的问题在目前还没有统一的答案！

我们还将会看到很多问题都涉及到逻辑中的怪圈，而由于计算理论已经触及了逻辑、信息的根本，所以把一些问题引向逻辑怪圈并不奇怪。

四、模拟

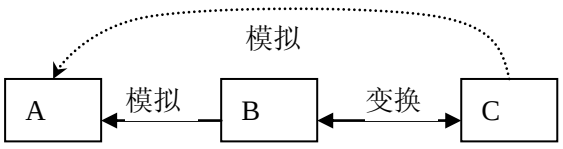
1、什么是模拟？

什么是模拟？又是一个基本的问题，爱因斯坦说过，越是基本的概念就越是难以刻画清楚。模拟这个概念就是一个很难说清的问题。

如果你站在一个朋友面前，冲着他做了一些鬼脸。那么他也会学着你的动作冲你做鬼脸，那么他就对你进行了模拟。

很明显，在你和你朋友之间存在着一系列的对应关系：你的手对应他的手，你的眼睛对应它的眼睛，你的嘴巴对应他的嘴巴……。而且你的手、眼睛、嘴巴做出来的动作也会对应他的手、眼睛、嘴巴做出来的动作。因而，模拟的关键是对应！如果集合 A 中的元素可以完全对应 B 中的元素，那么 A 就可以模拟 B。

仍然用你冲你的朋友做鬼脸的例子，假如这次你做出的鬼脸以及动作没有被他立即模仿而是被他用某种符号语言记录到了日记本上了。比如：“X 年 X 月 X 日，疯子 XX 冲我做了一个鬼脸：他伸出了左手食指放到了右眼下面往下拉他脸上的肉，并且吐出了他长长的舌头！”。过了 N 多天后，你的这位朋友掏出了日记本，按照上面的描述冲着大家做了这个鬼脸。很显然他仍然模拟了你当时的动作。那么，你朋友日记本上的那段对话描述是不是对你鬼脸动作的模拟呢？似乎答案是否，因为这段文字跟你没有半点相像。然而你的朋友正是根据这段描述才做出了对鬼脸动作的模拟。也就是说，他把那段文字翻译成了他的动作，而他这个动作就是对你的模拟。这个翻译的过程很显然就是某种信息的变换，我们完全可以把它理解为一个计算的过程，也就是可以用图灵机来实现的算法过程。所以，我们说日记本上的那段指令也构成了对你鬼脸动作的模拟，原因是这些信息也与你的鬼脸动作构成了对应。具体的，我们可以用下面的图表示：

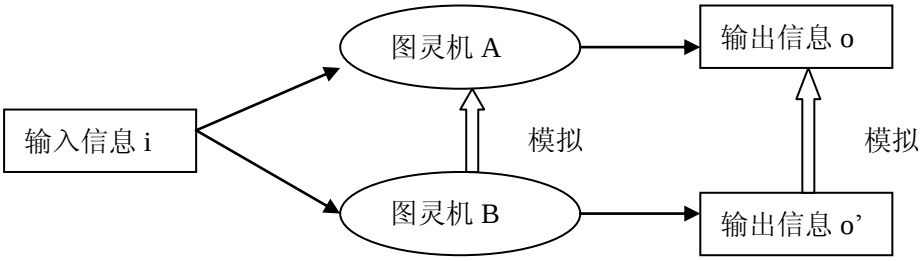


这里 A 是你的鬼脸动作，B 是你朋友做出来的鬼脸动作，C 是日记本上的描述。你朋友的动作 B 模拟了你的动作 A，而 B 的动作信息是通过执行 C 上的描述得到的，也就是说存在着一个从 C 到 B 上信息的变换。这样我们认为 C 也对 A 进行了模拟。

2、图灵机之间的模拟

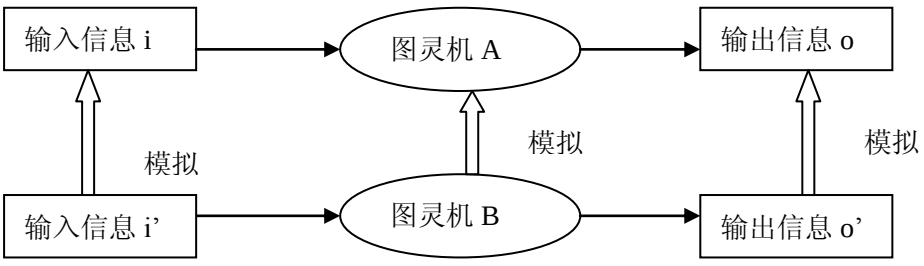
下面来考虑图灵机之间的模拟。按照前面的定义，一台图灵机包括：输入集合 I，输出集合 O，内部状态集合 S，程序规则表 T 四个要素。那么如果两个图灵机之间的这些元素都存在刚才说的对应关系，就认为两个图灵机可以相互模拟了。然而图灵机的功能是完成对输入信息进行变换得到输出信息的计算。我们关心的也仅仅是输入输出之间的对应关系。因而一台图灵机 A 如果要模拟 B 并不一定要模拟 B 中的所有输入、输出、内部状态、程序规则表这些元素，而只要在给定输入信息的时候，能够模拟 B 的输出信息就可以了。

因此，我们可以用下面的图来表示图灵机之间的模拟：



也就是说在给定相同输入信息的情况下，只要输出信息 o’能够模拟信息 o 就可以，也就认为 B 模拟了 A。而信息 o’对信息 o 的模拟又符合我们上面对一般集合之间模拟的定义。也就是说如果存在另外一台图灵机能够把信息 o’计算并映射成信息 o，就认为 o’模拟了 o。说白了也就是 o’可以与 o 不一样，但是只要你能用一个图灵机把 o’经过一系列运算变换到相同的 o，就认为 o’模拟了 o。因而也就是图灵机 B 模拟了图灵机 A。

进一步，我们可以假设 A 和 B 输入的信息也不一样，一个 i，另一个是 i’，那么如果 i 和 i’之间也存在着模拟对应关系的话，我们仍然认为 B 可以模拟 A。也就是下面的图：



有一点需要注意，如果 A 图灵机模拟了 B 图灵机，那么并不一定 B 图灵机可以模拟 A 图灵机。因为有可能 A 图灵机比 B 图灵机处理的信息更多。也就是说假如 B 能处理的信息就是 1，2，3，4，而 A 处理的信息除了这四个数之外，还有 5,6,7,8，那么显然当输入 1234 的时候 A 能够模拟 B，而当输入 5678 的时候 B 没定义了，不能完成任何操作。在这个时候 B 显然不能模拟 A 了。

3、计算等价性

讲了这么多关于模拟的知识有什么用呢？模拟的一个关键作用就是阐明什么是等价

的。比如为了完成加法运算，你写了一段程序，而我也写了另一段程序，虽然我们两个的程序可能完全不一样，然而只要我们两个程序之间能够相互模拟，也就是说只要给定两个数，我们都能正确的一模一样的算出它们的和，那么我们两个程序就是等价的！

具体地说，如果 A 能够模拟 B，并且 B 也能模拟 A，那么 A 和 B 就是计算等价的。计算等价性是非常强有力的，因为它揭示了在我们这个宇宙中某种非常普遍的规律。我们仍然用刚才说的加法算法为例子来说明。虽然计算两个数的加法的方法可能有无穷多种，也有可能用各种各样的计算机语言，什么 C,Basic,JAVA 等等来实现，更有可能奔跑在不同的计算机上，然而所有这些程序，这些计算的结果意义都是相同的。也就是说所有与加法运算算法计算等价的计算机程序都是一回事儿，因而加法算法这个东西是某种永恒而独立的！

看！我们在宇宙中找到了某种永恒性了，这种永恒性反映了宇宙规律中某种本质上的美！计算等价性就和能量守恒定律一样具有这种高级的对称性，我甚至觉得计算等价性要比能量守恒定律更加深刻！因为无论如何能量守恒定律仍然是刻画了物理系统的某种属性，而计算等价性则刻画的是非常广泛的信息系统之间的某种守恒和对称性，而一切系统都可以被抽象为信息系统，甚至是物质世界，所以，计算等价性是跨越所有系统之间的某种高级对称的、永恒的、美的东西。

为了进一步理解计算等价性的威力所在，我们不妨科幻一下。假设我们能够用计算机模拟某个人，比如说张三的思维过程了（也就是假设真正的人工智能可以实现了）。那也就是说我们可以用一个计算机软件 X 来完成对张三思维的模拟。这样，这个软件就会在一切与它具有计算等价性的程序甚至系统上实现张三这个人的思维过程！比如我们完全有可能让一大堆分子的碰撞来实现 X 这个软件，那么就会在这大堆分子碰撞的过程中完成对张三思维的模拟，也就是说张三这个人的意志蹦到了这一大堆分子系统中去了！更进一步，我们还可以找来足够多的人比如这个星球上所有的人来模拟那大堆分子的碰撞，从而完成软件 X 的计算。这意味着什么？意味着张三这个人的思维或者说意识在那群人的整体上突现了！很有可能，这些构成软件 X 的人都并没有意识到在他们上层的张三的意识的出现。更有趣的是，张三自己很有可能就在那一群人之中呢！

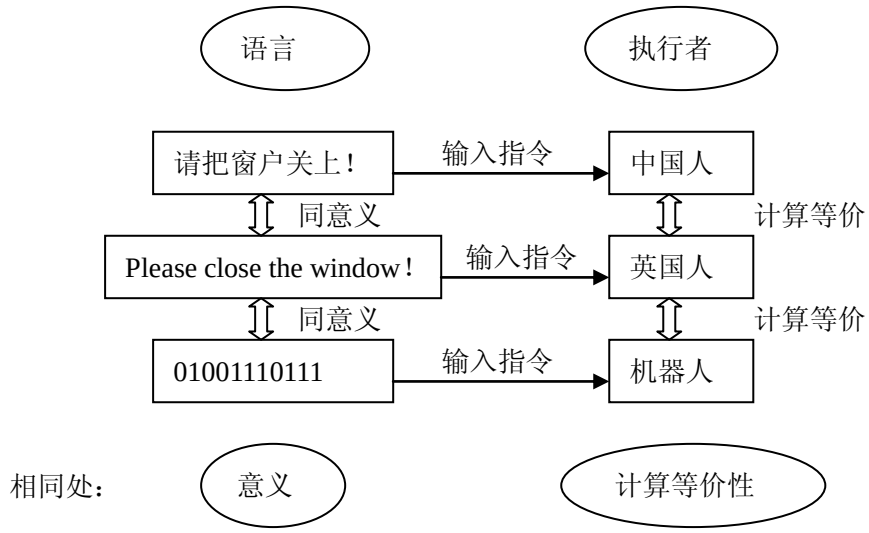
相信你已经能够参悟到了什么是计算等价性的威力了，那么我也相信你能够理解为什么说任何一台我们使用的计算机都不过是图灵机的翻版了。

4、意义*

考虑下面三句话：“请把窗户关上！”，“Please close the window！”，“01001110111”。这三句话分别说给不同房间中的三个人。第一句话告诉给一个中国人，于是他关上了窗户；第二句话告诉了一个英国人，他也关上了窗户；第三句话告诉的是一个机器人，他也关上了窗户。这三句话从表面看显然是完全不一样的，然而当它们让不同的人来听的时候，却达到了相同的最终结果：窗户被关上了。那么，我们自然会想，这三句话有何相同呢？显然，答案是他们的意义相同。然而什么又是意义呢？

真正回答意义的本质是一个很困难的问题，现在人们正在努力理解语义是什么。虽然我们仍没有完全回答这个问题，但是，不妨从图灵机、计算以及计算等价性的观点来考虑该问题。如果把中国人、英国人、机器人都看作是图灵机，而那三句话看作是对他们的输入信息，那么最终的结果就是图灵机计算的输出。这个时候我们看到三种结果是相同的。也就是说这些图灵机之间是可以相互模拟的。

考虑这三句话，显然它们都具有相同的意义。而根据前面的叙述，能够相互模拟的图灵机是具有相同的计算等价性的。因而描述听到关窗指令后并按照指令行事的图灵机具有相同的计算等价性。而这种计算等价性就好像是前面说到的加法规则一样是独立于计算系统、执行机构的。因而，我们能得到下面的图：



通过这个对比图，我们不难得出结论：所谓语言的意义，就是执行这个语言系统的计算等价性！

我们如何知道不同的语言表达了相同的意义呢？显然，我们只要有了翻译就可以明白“请把窗户关上”与“Please close the window”具有相同意义，而翻译所作的工作无非就是输入中文信息输出英文信息这样的信息转换工作，因而，也就是一个计算过程！

然而当不存在从一个语言到另外一个语言的翻译的时候，我们也并不能断定某一个符号序列对于固定的图灵机是否有意义。这就是说，我们虽然不能明白鸟叫是什么含义，但并不能否认它们的叫声可能有意义，因为只有鸟自己才能明白叫声的含义。

五、万能图灵机

1、编码

其实我说的这个“万能图灵机”就是计算机术语中的“通用图灵机”，英文是 Universal Turing Machine。而我之所以称之为“万能图灵机”完全是因为这个名字似乎听起来更加直观。

前面已经讲述了模拟的概念，那么自然会产生这样一个问题：存在不存在一台图灵机能够模拟所有其他的图灵机呢？答案是存在的。这种能够模拟其他所有图灵机的图灵机就叫做通用图灵机，也就是我们所说的“万能图灵机”。这种机器在图灵计算这个范畴内，是万能的！

“万能图灵机”会怎样工作呢？假如我把信息 x 输入到了图灵机 M 中，M 就能计算出一个结果 o。那么如果我把 x 和 M 的信息都输入给万能图灵机，那么万能图灵机也会输出 o，也就是万能图灵机可以模拟任何一台特殊的图灵机。这样的话我们就可仅仅通过改变输入 x 和 M 的值就能“改变”万能图灵机的程序规则了。因而也可以认为万能图灵机就是可以任意编程的。这里的改变两个字加上了引号，是因为事实上任何图灵机在诞生之后规则就不能改变了，因而我们能够改变“万能图灵机”的规则，仅仅是因为看上去是这样的，其实根本没有改变。

要说明为什么“万能图灵机”是存在的，以及它是怎样模拟其他任何图灵机的动作的，我们必须先要理解究竟怎样把任何一台图灵机输入到“万能图灵机中”，这就需要理解编码的概念。什么是编码呢？你可以理解为对某一堆事物进行编号就是编码。

其实我们每人每天都在跟编码打交道。每个人都有一个身份证，而这个身份证都有一

个 ID 号码吧？那么这个号码就是你的编号。上学的时候老师给我们每个人都分配一个学号也是编码。

26 个字母能够被编码，比如 a 对应 1，b 对应 2，……，这是显而易见的。然而任意一个英文单词都是可以被编码的则不那么容易一眼看出来。事实上，我们可以按照字典顺序把所有的单词都列出来。也就是说字母顺序越靠前，字符长度越短的单词排在前面，其次长单词，字母顺序靠后的单词就排在后面。比如一种可能的字典顺序：

a, about, an…, bad, be, behave…..

只要这样一排好序，我们就能给每个单词赋予一个数字，最简单的方法是，给第一个字母分配 1，第二个分配 2，……，因而我们就给所有的单词都编码了。

下面讨论任意一个图灵机能不能被编码。我们假设讨论的所有图灵机的输入集合都是仅有 0,1 两种，而它的输出也仅仅有 0,1,2,3 四个动作分别表示前移，后移，涂写 0，涂写 1。而内部状态数最多为 10000 个（总之足够多就可以了）。下面考虑程序。

假设图灵机的程序表为：

当前内部状态 s　输入数值 i　输出动作 o　下一时刻的内部状态 s'

2	1	0	3
1	0	3	2
3	0	1	1
…	…	…	…

那么我们可以把它写到一行中，这就是 2,1,0,3; 1,0,4,2; 3,0,1,1，注意用“,”分开了内部状态，输入数值，输出动作和下一时刻的状态，而用“;”分开了一行一行具体的程序。这样无论这个表有多大，我们都可以把它写成这样的一个字符串。这个字符串就相当于一个英文单词，这就是对该图灵机程序的一个描述。同理，其他的图灵机也能够得到这样的一个单词描述，那么我们再用字典序的方法对这些描述进行编码，也就得到了对所有图灵机的编码。

如果一台图灵机的编码是 M，它读入的信息是 x，这样只要把 M 和 x 用“.”号隔开的方法分开作为数据输入到“万能图灵机”中，运用特殊的算法，这个万能的机器就能得出对 M 计算 x 的模拟结果了。事实上可以由定理证明万能图灵机对于任意的编码都是存在的，在这里我们就不叙述证明过程了。

2、自食其尾

既然“万能图灵机”能够模拟任何一台图灵机的动作，那么它能不能模拟它自己的动作呢？答案是肯定的。我们首先看到“万能图灵机”也是图灵机，也有固定的输入、输出、状态的集合、固定的程序，因而它也能被编码。于是我们就可以把它自己的编码信息输入给它自己了。这就好像一条蛇咬到了自己的尾巴。会发生什么呢？自食其尾就会产生怪圈，虽然我们现在还没有看到任何不好的征兆，然而在下一节里面，我们将看到这种怪圈会诞生什么样的结论。而且我们也会看到，其实这个怪圈是和康托尔对角线法则、哥德尔定理有关的。

图灵机一旦能够把程序作为数据来读写，就会诞生很多有趣的情况。首先，存在某种图灵机可以完成自我复制！事实上，计算机病毒就是这样干的！我们简单说明一下，这个特殊的图灵机是如何构造的。我们假定，如果一台图灵机是 X，那么它的编码就记为<X>，这样能够自我复制的图灵机 T 的功能是，把 T 的编码<T>写到纸带上输入到“万能图灵机”，

那么“万能图灵机”就能根据读入的<T>，在纸带上再次输出<T>的一份拷贝<T>’，并且<T>=<T>’。下面就来大概解释如何构造这样的 T。首先 T 由两部分构成 AB。第一部分 A 的功能是指导“万能图灵机”把 B 的编码<B>原封不动的打印到纸带上，这个时候纸带上就有了<B>，如果这个时候你想用同样的方法打印<A>到纸带上是不行的，因为会出现循环定义。然而 B 可以这样做，读入纸带上的信息 X，生成能够打印 X 的图灵机：p(X)的编码<p(X)>打印到纸带上，并把 X 和<p(X)>的内容前后调换，有定理保证这样的图灵机是存在的。这样当 B 读到纸带上的信息<B>之后就会打印出能够打印<B>的图灵机的编码也就是<A>了，然后把<A>和<B>位置对换就构成了<AB>也就是<P>，所以 P 把自己进行了一次拷贝。初看起来，这种自我复制的程序是不可能的，因为这包含了无穷无尽的怪圈。P 要能产生它自己<P>就意味着 P 中至少包含了一个<P>，而这个<P>中又包含了至少一个<P>……，最后 P 必然是一个无限大的程序，然而我们却能够证明 P 是可能的。

有了“万能图灵机”还能得到很多有趣的结论，比如假设有一大群图灵机，让它们彼此之间随机的相互碰撞，当碰到一块的时候，一个图灵机可以读入另一个图灵机的编码，并且修改这台图灵机的编码。那么这样一个图灵机“汤”中会产生什么呢？圣塔菲研究所的芳塔娜已经研究了这个实验，并得出了惊人的结论：在这样的系统中会诞生自我繁殖的、自我维护的类似生命的复杂组织，而且这些组织能进一步联合起来构成更大的组织！

六、停机问题

1、死循环

在进行正式讨论之前，我们先来看看一个非常简单的猜硬币游戏。

假如我的两个拳头中一个攥着一枚硬币，另一个没有，然后让你猜是哪一个？于是你告诉我左手中有。这时候我不会把手张开，而是背过身去做一番手脚，然后把拳头伸过来，张开手！哈，你错了吧，硬币在右手中！大概傻子都能看出来我的伎俩之所在！不用说，采用这种方法我保证百战百胜。因为我总是等你说出来哪个手有硬币之后再动态的改变我的策略。所以，这改变之后的状态就已经不是你猜的了。

大概你会觉得不可思议：其实图灵停机问题就是一个类似该游戏的原理！

下面我们来看看图灵停机问题是怎么回事儿。还记得我们前面提到的可怜的“小虫模型”么？当时我们就提出来一个问题：会不会存在某种聪明的数学，只要检查一下小虫的程序，而不用执行该程序就能够让我们预言小虫什么时候会陷入“傻循环”，而无休止的重复前面的动作？我们不妨假设这样的“聪明的数学”就是一个程序，它可以读入小虫的程序规则，然后吐出一个答案说明小虫会死循环，并且在多少多少步之后。那么这样的“聪明程序”存在么？

因为前面已经说了，小虫模型就是一个图灵机，那么“聪明程序”的问题就变成了正规的描述：存在不存在一个程序比如说 P，能够判断出任意一个程序 X 是否会在输入 Y 的情况下陷入死循环？我们不妨设 P(X,Y)表示 P 判断程序是 X，数据是 Y 的结果。如果存在死循环，那么 P(X,Y)就输出一个 yes。如果不存在死循环，那么 P(X,Y)就输出一个 no。我们的问题就是这样的 P(X,Y)存在么？这就是停机问题。所谓的某个程序 X 在输入 Y 上停机就是说 X 不存在着死循环，反过来如果不停机就是存在着死循环，因而这里停机和死循环是一回事儿。那么，这种判断停机问题的程序 P 存在么？

答案是不存在的。下面我可以证明我的这个结论。

我们不妨假设程序 P 存在。那么我们可以根据 P 设计一个新的程序 Q 如下：

X 是任何一段程序的编码：

Program Q(X){

```
    m=P(X,X)
    do while (m=no)
        …
        …
     end do
    if m=yes then return
}
```

这段程序通俗来讲就是：输入任何一段程序 X，调用函数 P(X,X)并得到返回值 m，如果 m=no，也就是说根据 P 的定义，P 判断出程序 X 作用到它自己身上 X 不存在死循环。那么 Q 就不停的做 do while 和 end do 之间的语句。如果 m=yes，我们知道这表示 P 判断出程序 X 在 X 上存在死循环。就返回，结束该函数。

我们可以看到，这样定义的函数 Q(X)是没有问题的。下面就进入关键时刻了：我们问 Q 这个程序作用到 Q 自身的编码上也就是 Q(Q)会不会死循环呢？当然我们可以运用强有力的函数 P(Q,Q)来计算这个问题。

假设 Q(Q)会发生死循环，那么 P(Q,Q)就会返回 yes。然而根据 Q 函数的定义，把 X=Q 代入其中会发现由于 P(Q,Q)返回的是 yes，也就是 m=yes，因此 Q 函数会马上结束，也就是程序 Q(Q)没有发生死循环。然而如果假设 Q(Q)不发生死循环，那么 P(Q,Q)应该返回 no，这样根据 Q 函数的定义，把 X=Q 代入 Q(Q)之中会得到 m=no，这样程序就会进入 do while 循环，而这个循环显然是一个死循环。因而 Q(Q)发生了死循环！这又导致了矛盾。

无论 Q(Q)会不会发生死循环，都会产生矛盾，然而哪里错了呢？答案只能是最开始的前提就错了，也就是说我们最开始的假设 P(X,Y)能够判断任意程序 X 在输入 Y 的时候是否死循环是错误的！也就是说这样的程序 P(X,Y)不存在！

2、如何理解

也许你会感觉整个论证过程有些怪异，为什么不存在这种 P(X,Y)程序呢？而上面的论证过程中仅仅说 P(X,Y)当作用到 P(Q,Q)上时会产生矛盾。似乎并不能说明 P 作用到其他程序上不能判断是否死循环。比如你可以考虑编写这样一段程序，当一发现某个程序 do while(T)，这里 T 总是为真，这样的语句出现的时候就判断这个程序有死循环。这显然是可能的。但问题的关键是，你假设了 P(X,Y)能够判断任意的一个程序是否死循环！最关键的就是这“任意程序”上了。因为假如你已经按照刚才提到的判断是否有 do while(T)语句的方法写出了一个程序 P 来判断某程序是否死循环，那么我就会根据你这个程序 P 再构造出一个程序 Q，就是利用上面提到的论证方法，我们不妨写成 Q_P(这里下标 P 的含义表示根据你的程序 P 而构造的 Q)。这样你的 P 在遇到了 P(Q,Q)这样的怪东西的时候无能为力了！

可能你还不服输，于是你又改进了你的程序变成了 P’，这个时候 P’能够判断包含了 Q_P 这个程序时候的所有程序情况了。那么我又会根据你的新程序 P’来构造出一个更新的 $Q_{P'}$，你的程序 P’仍然不能判断，当然你还可以构造 P”,P”’,……，我也会跟着构造 $Q_{P''},Q_{P'''}$，……，总而言之这个过程是无穷的！因为我总在你之后构造程序，所以你是水我是船，水涨船高，我总能比你高一级别！

这很像刚开始叙述的那个猜硬币的游戏。你想猜对我的硬币，就必须告诉我一个答案是左手还是右手，然而关键问题是我总能根据你做出的答案动态调整，使得你永远也猜不对！停机问题也是如此，我总能根据你的程序 P 来构造你的 P 判定不出来的问题 Q，我总会赢！很简单，因为你总要在我之前构造好 P 呀，就相当于你总要先说出硬币在哪个手！

3、对角线删除方法

我在开始的时候就提到了图灵停机问题、哥德尔定理等等都来源于康托尔的对角线删除法则。那么，下面我们就来看看，如果运用对角线删除法则如何证明图灵停机问题呢？采用这种全新的证明思路，也许你会更加清楚地认识到停机问题的本质。

问题没有变，是否存在一个程序 P(X,Y)判断出来任意一个程序 X 当输入 Y 的时候是否有死循环，或者说是否停机。

我们仍然用反证法，假设这样的 P(X,Y)是存在的。而我们知道程序 X 本身是可以被编码的。也就是可以为所有的程序进行编号：1，2，3，……，而数据 Y 本身也是这样的编号 1，2，3，……，因而我们就可以把每一对 X 和 Y 的组合排列在一张表上。比如列表示的是数据 Y，而行表示的是程序 X，这样 P(X,Y)的值也就是 yes 或 no 就可以写在第 X 行第 Y 列的对应位置上，表示 P(X,Y)判断出的 X 作用在输入 Y 上是否会死循环的结果。比如下面的表就是一个示例：

	1	2	3	4	5	6	……	Y	……
1	yes	no	no	yes	yes	no	……	yes	……
2	no	no	yes	yes	no	yes	……	no	……
…	……								
X	no	yes	no	yes	yes	no	……	yes	……
…	……								

到这里没有发生什么毛病。我们知道上表中的每一个行都表示一个确定的程序作用到不同的数据上所得到的结果。比如程序 X 作用在 1,2,3,4,……,Y，……上给出的结果就是一个序列：

no,yes,no,yes,yes,no,……,yes，……

下面我们把上表对角线上的元素挑出来。也就是专门找那些第 1 行第 1 列，第 2 行第 2 列，……这样的元素就可以得到一个序列：

yes,no,no,yes,……

而根据这个序列我们完全可以构造这样一个反序列：

no,yes,yes,no,……

也就是说这个序列在每一个位上都与前一个对角线序列不同!这个序列就称为对角线删除序列。那么我问，这个对角线删除序列是否在我们表中的某一行上呢？答案是否定的！这个序列必然不在表中。因为它的第一个元素与 1,1 不同，第二个元素与 2,2 不同，……。换一种方式解释就是：假设对角线删除序列的确在表上列出来了，那么这个序列必然在某个固定的行，比如说就在第 1001 行。那么我们就要考虑这第 1001 行，第 1001 列的元素，我们知道根据序列的构造方法(1001,1001)对应表中的元素是与序列上的这个元素不同的!因而产生了矛盾，也就是说这个构造出来的对角线删除序列不在表中!

我们完全可以设计出一段程序 Q 使得 Q 作用在 1,2,……,X,……上产生的序列就是对角线删除序列，而对角线删除序列不在表中就意味着程序 P 并不能判断出程序 Q 作用在任意输入上是否停机。其实这里的程序 Q 就是前一节论证停机问题的程序 Q。

用对角线删除的证明方法来看究竟哪里出错了呢？错误就出在我们假设程序 P 能够得到这样一张完整的表，这张表对所有的程序计算得到是否停机的答案。

4、意味着什么？

我们已经看到了，的确存在着一类问题我们人类能构造出来，而图灵机是不能解的，。我们知道图灵机不能解的问题也就是一切计算机不能解的问题，因而这类问题也叫做不可计算的。因此，必然存在着计算机的极限。实际上，运用我们前面叙述的计算等价性原理，有很多问题都可以被归结为图灵停机问题，也就是说图灵停机问题揭示了宇宙中某种共性的东

西，所有那些计算机不能解决的问题从本质上讲都和图灵停机问题是计算等价的。比如在最开始我们就提到的希尔伯特第 10 问题就是一个典型的不可计算问题！还有很多问题是不可计算的，尤其是那些涉及到计算所有程序的程序。比如是否存在一个程序能够检查所有的计算机程序会不会出错？这是一个非常实际的问题。我们都知道计算机程序特别容易犯错误，为了检查出某段计算机程序的错误，我们人类就必须对这个程序进行人工的检查。那么能不能发明一种聪明的计算机软件，输进去任何一段计算机程序，这个软件就会自动帮你检查输入的程序是否有错误？答案仍然是不存在的，其实这个问题可以被证明和图灵停机问题实质上是一样的！于是我们的梦想又破灭了！

图灵停机问题也和复杂系统的不可预测性有关。我们总希望能够预测出复杂系统的运行结果。那么能不能发明一种聪明的程序，输入进去某个复杂系统的规则，输出的是这些规则运行的结果呢？从原则上讲，这种事情是不可能的。它也是和图灵停机问题等价的。因而，我们得出来的结论就是：要想弄清楚某个复杂系统运行的结果，唯一的办法就是让这样的系统实际运作，没有任何一种计算机算法能够事先给出这个系统的运行结果。你很有可能不同意我的观点，因为毕竟我们能够发明很多预测复杂系统的方法，而不需要一定要让复杂系统去真实的运作。但是，你还是没有理解我这里的问题。我们强调的是不存在一个通用的程序能够预测所有复杂系统的运行结果，但并没有说不存在一个特定的程序能够预测某个或者某类复杂系统的结果。那么这种特定的程序怎么得到呢？显然需要我们人为地编程得到！也就是说存在着某些机器做不了的事情，而人能做。这似乎为人工智能的崇拜者给以了沉重的打击！

人工智能真的是不可能的么？彭罗斯曾经写过一本科学明著：《皇帝新脑》来论证人工智能的不可能性。它所运用的方法就是我们上面的逻辑。因为对于任何一个人工智能程序来说，总存在着它解决不了的问题！但是似乎我们人类却不受这种限制，我们总是能够发现一个程序是否有死循环，总是能够找到对某类复杂系统预测的方法，并且我们还能构造出来图灵停机问题这样的问题。然而事实并没有那么简单，反对者马上就会论证到，其实针对某一个具体的人，比如说就是彭罗斯，我们也能够运用前面的方法构造出一个彭罗斯自己不能解的问题！然而事实情况下要构造彭罗斯不可解的问题太麻烦了，而我们只是说原则上讲这种问题是存在的！因而计算机超越不了的问题，人自己也超越不了，所以说人工智能是可能的！

看看上面提到的两方面论证似乎都很有道理，究竟哪个正确呢？真的会存在某个人不可解的类似图灵停机的问题么？其实要想彻底回答这个问题就相当于问超越图灵计算的限制是否可能？如何超越图灵机停机问题呢？下面我们将详细讨论一下这个问题。

5、超越图灵计算*

我们仍然用那个猜硬币的游戏为例来说明。

在进行了几轮猜硬币的游戏之后，你已经很恼火了，认为这样的游戏不公平。于是你想了一个妙招来对付我：每当我让你说硬币在哪个手中时，你先胡乱的说一个答案，比如左手。这个时候我会根据你的答案动态调整把硬币放到了右手中。这个时候你赶紧抢着说，不对，我猜你的硬币在右手！我没办法只能再次调整策略把硬币放到了左手。你又赶快说：是在左手！……。就是这样，你也学会了我的方法，根据我的策略不断调整你的策略从而让我不可能赢你。能不能把这种方法用到超越图灵停机问题呢？

前面我们已经看到了类似这样的过程。假如你写出了一个程序 P 能够判断所有程序是否停机，那么我就能够构造一个程序 Q 是你的程序判断不了的。这个时候还没有结束，你又根据我的 Q 构造了新的程序 P’，然而我又能构造一个程序 Q’仍然让你的程序 P’解决不了。但是你没有结束，又构造了新的程序 P’’，我于是又构造 Q’’……。

乍一看，似乎这个过程并不能说明任何问题。原因很简单，我要求的是构造一个固定的

程序 P 判断出所有程序是否停机。而你给我的并不是一个具体的实实在在的程序，而是一个不断变化的捉摸不定而虚无飘渺的程序序列!并且你的这些总在变化的程序序列总是要根据我构造的程序才会确定改变!

首先一点值得肯定的是，运用这种方法，我们的确能够超越图灵计算了，只要反复不停的变换我们的程序就不可能找出它不能解的问题。然而，另一方面又会让我们很失望：这样的变换过程并不能给出一个实实在在的程序来!我们拥有的仅仅是不断改变的程序序列，而不是一个实际存在的程序!

这正是问题的关键所在:要想彻底超越图灵计算的限制,我们必须要放弃程序的实在性。也就是说程序在每时每刻都要变化成不是它自己了。那么这样的一个不断变化得不是它自己的怪东西存在么?

几千年的人类科学一直在研究实实在在的东西。无论是原子、分子还是计算机程序，它们必须是一个实实在在存在的个体，在这种前提下科学才能够对它进行研究!如果当我们研究它的时候，它已经变得不是它自己了，那么科学就对它无能为力了。然而，我不禁要提出这样的问题：真的一切都是固定不变的存在着么，有没有某种东西在每一时刻都在变得不是它自己了呢?

这个问题似乎是一个古老的哲学问题了记得赫拉克里特就曾经提到过:一个人不能两次踏入同一条河流。我想他说的正是这样的问题：因为河流在每时每刻都不再是它自己了。河流是一大群流动的水滴构成的整体，在每时每刻这些水滴都在不停的运动、流逝，因而当你两次踏入这条河的时候，所有的水滴可能都不一样了，那么我们怎么能说这些水滴构成的整体还是同一条河呢?

再考虑我们人自己。你很可能拿着一个你 3 岁时候的照片兴奋的对你的朋友说：“看，我 3 岁的时候多可爱呀!”。然而你这句话意味着什么呢？意味着照片反映的 3 岁时的你和现在的你是同一个个体！然而，3 岁的你和现在的你是多么不同呀！我们知道，你无疑就是一大堆细胞构成的一个整体。而基本生理学知识告诉我们，实际上人体的所有细胞每隔大约 4 年就会因为新陈代谢的作用全部更新一遍。也就是说，你的细胞全被掉了包了，更何况 3 岁时候的你和现在的你差了多少个4年呀？那凭什么说那个3岁时候的你就是现在的你呢?

这个问题看似玄学，不过我认为现在我们的确应该认真对待该问题了。尽管从分析的角度来说 3 岁的你和现在的你的确不是一个个体，然而常识告诉我们，这两个你的确都是同一个人！那就意味着，你这个个体并不是一成不变的一些固定的细胞，而是一个每时每刻都在变化，都在更新的一个一大堆细胞组成的构形。这个构形在每时每刻都要利用更新的一大堆细胞去维持自己的存在！我们得到了什么？和我们前面叙述的超越图灵机的讨论结合起来，就发现，原来人还有赫拉克里特的河流这种东西刚好就满足那种超越图灵计算的要求。也就是说人还有赫拉克里特的河流在每时每刻都在不停的更新它自己从而变得不是它自己了。那么很有可能，某一种做类似变化的个体的变化规律就是不停超越它自己的图灵停机程序，这样的虚幻的个体就真的能够超越图灵计算了!

总结前面的讨论，我们不难给出结论，一个固死的能够被写出就不再变化的程序不可能超越图灵计算的限制，然而如果一个程序每时每刻都已经变化得不是它自己了，这个程序就能够超越图灵计算。联系到人这个个体，我们能得到：因为每时每刻的人都已经由于细胞的变化而变得不再是它自己了,所以人是超越图灵计算的!还记得在前面我提到的一个问题么：“人脑的信息处理过程能不能被表示成固定的程序呢？”。我这里的答案就是否定的！也就是说人脑信息处理的过程并不是一个固定的程序!如何制造真正的人工智能呢？很显然，我们不能用一个简单的程序来构造，而必须是利用其它的方法，这个方法是什么呢？现在还没有结论!

七、悬而未决

到此，我已经把全部的有关图灵机、可计算理论、停机问题的一些重要概念介绍完了。然而在计算理论这个领域里还有很多重要的问题没有介绍。但我不得不根据我的兴趣进行取舍。在整个介绍过程中，一方面我介绍了人们已经得出来的结论，另一方面，我尽量把一些没有解决的问题展现给大家。回忆起来，这些悬而未决的问题包括下面几个：

1、 是否一切的信息处理过程都具有固定的程序呢？人脑有固定的程序么？

2、 如何用计算机程序进行归纳？能对所有事物进行一劳永逸的归纳算法是否存在？

3、 什么是意义，更确切的，什么是语义？

4、 图灵停机问题是不可超越的么？

5、 人工智能是否可能？

还有很多问题在本文中没有提出来，然而我认为也是相当重要的。例如，我们如何用图灵机模型表示一般的学习过程？若干小的图灵机是如何自动的构造出更大的图灵机的（这就是万事万物自组织的过程）？生命的目的性如何用图灵机模型表示？

另外最近的计算主义已经把宇宙中一切的过程都归结为计算过程了，也就是说到处都是图灵机正在做运算。那么我们能不能从图灵机的角度探讨时间和空间的本质呢？我们知道，计算理论另外一大类问题就是探讨计算的时间和空间复杂度，那么这种计算的时间和空间与我们这个宇宙的时间和空间有什么关系呢？

我希望，在新世纪的今天，人们会最终解答这些问题！

书写之中不免有很多错误和遗漏，也可能会存在很多逻辑漏洞，如果你发现了，请告诉我，我不禁感谢！

2004 年 7 月

集智俱乐部：http://www.swarma.org