图灵的光环

仿佛全世界的人都知道，图灵（Alan Turing）是个天才，是他创造了计算机科学，是他破解了德国纳粹的 Enigma 密码。他被叫做“计算机之父”。由于他的杰出贡献，计算机科学的最高荣誉，被叫做“图灵奖”。然而根据自己一直以来对图灵机等计算模型的看法，加上一些历史资料，我发现图灵本人的实际成就，相对于他所受到的崇拜，其实相差甚远。

由于二战以来各国政府对于当时谍报工作的保密措施造成的事实混淆，再加上图灵的不幸生世所引来的同情，图灵这个名字似乎拥有了一种扑朔迷离的光环。人们把很多本来不是图灵作出的贡献归结在他身上，把本来很平常的贡献过分地夸大。图灵的光环，掩盖了许多对这些领域做出过更加重要贡献的人。

图灵传

2012年，在图灵诞辰一百周年的时候，人们风风火火的召开各种大会，纪念这位“计算机之父”，很多媒体也添油加醋地宣传他的丰功伟绩。还有个叫 Andrew Hodges 的人抓住这个时机，推销自己写的一本传记《Alan Turing: The Enigma》。这本书红极一时，后来还被改编成了电影。

这本传记看似客观，引经据典，字里行间却可以感受到作者对图灵个人的膜拜和偏袒，他在倾心打造一个“天才”。作者片面地使用对图灵有利的证据，对不利的方面只字不提。仿佛图灵做的一切都是有理的，他做的不好的地方都是因为别人的问题，或者风水不好。提到别人做的东西，尽是各种缺陷和局限性，不是缺陷也要说成是缺陷；提到图灵的工作，总是史无前例，开天辟地的发明。别人先做出来的东西，生拉硬拽，硬要说成是受了图灵的“启发”，还怪别人没有引用图灵的论文。这让你感觉仿佛别人都在抄袭图灵伟大的研究成果，都在利用他，欺负他似的。如果你不想花钱买书，可以看看同一作者写的一个图灵简要生平，足以从中感受到这种倾向。

我写这篇文章的很大一部分原因，就是因为这本传记。作者对图灵贡献的片面夸大，对其他一些学者的变相贬低，让我感到不平。图灵在计算机界的名声，本来就已经被严重的夸大和美化，被很多人盲目的崇拜。现在出了这本传记和电影，又在人们心中加重了这层误解。所以我觉得有必要澄清一些事实。

密码学

很多人提到二战 Enigma 密码的故事，就会把功劳一股脑地归到图灵头上，只字不提其他人。其实破解 Enigma 密码是很多人共同努力的结果，图灵只是其中的一员。这些人缺少了任何一个，都可能是灾难性的后果。其中好些人的想法早于图灵，启发过图灵，设计的东西比图灵的先进，却很少有人听说过他们的名字。图灵有自己的贡献，但最后说起来倒好像是他单枪匹马拯救了大家，这是不公平的。

最初破掉 Enigma 密码的其实不是英国人，而是波兰人。波兰人不但截获并且仿造了德国人的 Enigma 机器，而且发现了其中微妙的漏洞，发明了一种用于解密的机器叫做 BOMBA，发明了一种手工破解的方法叫做 Zygalski sheets。BOMBA 可以在两个小时之内破解 Enigma 密码。波兰人一声不吭地窃听了德国人的通信长达六年半，最后在二战爆发前夕把这技术送给了英法盟友。

BOMBA 的工作原理就是同时（ 并发）模拟好几个 Enigma 机器，这样可以加速猜出秘钥。最开头这样还行，但后来德国人改进了 Enigma 机器，把可选的齿轮从 3 个增加到了 5 个，使得秘钥的空间增大了 60 倍。理论上 BOMBA 只要运转 60 倍多的 Enigma 机器，就可以破解这增大的解空间，然而那已经超出了波兰的物资和人力。再加上德国人就要打过去，所以波兰只好请英法盟友帮忙。

图灵最重要的贡献，就是改进波兰人的 BOMBA，设计了一个更好的机器叫 BOMBE。BOMBE 比起 BOMBA 并没有质的飞跃，只不过 BOMBE 同时模拟的 Enigma 机器更多，转的更快。另外它加入了一些“优化”措施，尽早排除不可行的路径，所以速度快很多。图灵最初的设计，要求必须能够事先猜出很长的文本，所以基本不能用。后来 Gordon Welchman 发明了一种电路，叫做 diagonal board，才使 Bombe 能够投入实用。关于 Gordon Welchman 的故事，你可以参考这个 BBC 纪录片。

在 Bombe 能够投入使用之前，有一个叫 John Herivel 的人，发现了一种特殊的技巧，叫做 Herivel tip，这种技术在 Bombe 投入使用之前几个月就已经投入实用，破解掉很多德军的消息，立下汗马功劳。如果 Herivel tip 没有被发明，盟军可能在 1940 年 5 月就已经战败，BOMBE 也就根本没机会派上用场。

同时在 Bletchley Park，还诞生了一台大型可编程电子计算机 Colossus，它是由一个叫 Tommy Flowers 的工程师设计的。Colossus 不是用来破解 Enigma 密码的，而是用于破解 Lorenz SZ-40。那是一种比 Enigma 更先进的密码机器，用于发送希特勒的最高指令。

德国人后来又改进了他们的通信方式，使用了一种具有四个齿轮的 Enigma 机器。这大大的增加了破解的难度，普通的 Bombe 机器也破不了它了。后来是 Harold Keen 设计了一个叫做 Mammoth 的机器，后来加上美国海军的帮助，制造了更快的 Bombe，才得以破解。

所以你看到了，所有这些人的工作加起来，才改善了二战的局面。波兰人的 BOMBA，已经包含了最重要的思想。图灵的工作其实更多是量的改进，而不是质的飞跃。现在很多人喜欢跟风，片面的夸大图灵在其中的作用，这是不对

的。如果你对 Enigma 机器的技术细节感兴趣，可以参考这两个视频：[视频1][视频2]。

理论计算机科学

图灵被称为“计算机之父”，计算机科学界的最高荣誉叫做“图灵奖”（Turing Award）。然而如果你深入的理解了计算理论和程序语言理论就会发现，图灵对于理论计算机科学，并没产生长远而有益的影响。在某种程度上说，他其实帮了一个倒忙。图灵的理论复杂不堪，给人们造成很大的误导，阻碍了计算机科学的发展。而且他对于发表论文，对待研究的态度让我怀疑，我觉得图灵本人其实就是当今计算机学术界的一些不正之风的鼻祖。

图灵机和 lambda 演算

绝大部分计算机专业的人提到图灵，就会想起图灵机（Turing Machine）。稍微有点研究的人，可能知道图灵机与 lambda 演算（lambda calculus）在计算能力上的等价性。然而在“计算能力”上等价，并不等于说它们具有同样的价值，随便用哪个都无所谓。科学研究有一条通用的原则：如果多个理论可以解释同样的现象，取最简单的一个。虽然 lambda 演算和图灵机能表达同样的理论，却比图灵机简单，优雅，实用很多。

计算理论（Theory of Computation）这个领域，其实是被图灵机给复杂化了。图灵机的设计是复杂而缺乏原则的。它的读写头，纸带，状态，操作，把本来很简单的语义搞得异常复杂。图灵机的读写两种操作同时发生，这恰好是编程上最忌讳的一种错误，类似于C语言的 i++。图灵机是如此的复杂和混淆，以至于你很难看出它到底要干什么，也很难用它清晰地表达自己的意思。这就是为什么每个人上“计算理论”课程，都会因为图灵机而头痛。如果你挖掘一点历史，也许会发现图灵机的原型，其实是图灵母亲使用的打字机。用一台打字机来建模所有的计算，这当然是可行的，然而却复杂不堪。

相比之下，lambda 演算更加简单，优雅，实用。它是一个非常有原则的设计。Lambda 演算不但能清晰地显示出你想要表达的意思，而且有直接的“物理实现”。你可以自然的把一个 lambda 演算表达式看成是一个电子线路模块。对于现实的编程语言设计，系统设计，lambda 演算有着巨大的指导和启发意义。以至于很多理解 lambda 演算的人都搞不明白，图灵机除了让一些理论显得高深莫测，还有什么存在的意义。

历史的倒退

图灵机比起 lambda 演算来说，其实是一个历史的倒退。1928年，Alonzo Church 发明了 lambda 演算（当时他25岁）。Lambda 演算被设计为一个通用的计算模型，并不是为了解决某个特定的问题而诞生的。1929 年，Church 成为普林斯顿大学教授。1932 年，Church 在 Annals of Mathematics 发表了一篇论文，纠正逻辑领域里几个常见的问题，他的论述中用了 lambda 演算。1935 年，Church 发表论文，使用 lambda 演算证明基本数论中存在不可解决的问题。1936 年 4 月，Church 发表了一篇两页纸的“note”，指出自己 1935 年那篇论文可以推论得出，著名的 Hilbert “可判定性问题”是不可解决的。

1936 年 5 月，当时还在剑桥读硕士的图灵，也写了一篇论文，使用自己设计的一种“计算机器”（后来被叫做图灵机）来证明同一个问题。图灵的论文投稿，比 Church 最早的结论发表，晚了整整一年。编辑从来没见过图灵机这样的东西，而且它纷繁复杂，远没有 lambda 演算来得优雅。就像所有人对图灵机的第一印象一样，编辑很难相信这打字机一样的操作方式，能够容纳“所有的计算”。编辑无法确定图灵的论述是否正确，只好找人帮忙。Church 恐怕是当时世界上唯一能够验证图灵的论文正确性的人，所以一番好心之下，编辑写了封信给 Church，说：“这个叫图灵的年轻人很聪明，他写了一篇论文，使用一种机器来证明跟你一样的结果。他会把论文寄给你。如果你发现他的结果是正确的而且有用，希望你帮助他拿到奖学金，进入普林斯顿大学跟你学习。”

图灵就是这样成为了 Church 的学生，然而图灵心高气傲，恐怕从来没把 Church 当成过老师，反倒总觉得 Church 抢先一步，破坏了自己名垂青史的机会。跟 Church 的其它学生不一样，图灵没能理解 lambda 演算的精髓，却认为自己的机器才是最伟大的发明。进入 Princeton 之后，图灵不虚心请教，只是一心想发表自己的论文，想让大家对自己的“机器”产生兴趣，结果遭到很大的挫折。当然了，一个名不见经传的人，做了个怪模怪样的机器，说它可以囊括宇宙里所有的计算，不被当成民科才怪呢 ;)

1937 年，在 Church 的帮助下，图灵的那篇论文（起名为《Computable Numbers》）终于发表了。Church 还是很器重图灵的，他把图灵的机器叫做“图灵机”。不幸的是，论文发表之后，学术界对此几乎没有任何反响，只有两人向图灵索取这篇论文。图灵当然不爽了，于是后来就到处推销自己的图灵机，想让大家承认那是伟大的发明。有了一个锤子，看什么都是钉子。后来每到一个地方，每做一个项目（见下一节），他都想把问题往自己那篇论文和图灵机上靠，东拉西扯的想证明它的价值，甚至称别人发明的东西全都是受到了图灵机的启发…… 经过人们很长的时间的以讹传讹之后，他终于成功了。

图灵当年的作法，其实跟当今计算机学术界的普遍现象差不多。我想发表自己的想法 A，结果别人已经发表了 B，解决了 A 要解决的问题，而且还比 A 简单和清晰。怎么办呢？首先，我声明自己从没看过 B 的论文，这样就可以被称为“独立的发现”。然后，我证明 A 和 B 在“本质”上是等价的。最后，我东拉西扯，挖掘一下 B 的局限性，A 相对于 B 在某些边沿领域的优势…… 这样反复折腾，寻找 A 的优势，总有一天会成功发表的。一旦发表成功，就会有人给我唱高调，没用的东西也要说成是有用的。他们会在 A 的基础上发展他们自己的东西，最后把我推崇为大师。那发表更早，更简单优美的 B，也就无人问津了。胜利！

现在不得不说一下《图灵传》对此的歪曲。Church 的论文发表，比图灵的论文投稿还早一年，而且 Church 使用了比图灵机更简单优雅的计算模型。Church 的成果本来天经地义应该受到更多的尊重，到头来作者却说：“… and

Turing was *robbed* of the full reward for his originality”（见第 3 节“The Turing machine”）。让人感觉貌似是 Church 用不正当的手段“抢走”了图灵的“原创性”一样。本来没有什么原创性，还丑陋复杂，所以何谈抢走呢？我怎么觉得恰恰相反，其实是图灵抢走了 Church 的原创性。现在提起 Hilbert 可判定性问题，可计算性理论，人们都想起图灵，有谁还想得起 Church，有谁知道他是第一个解决了这问题的人，有谁知道他用了更优美的办法？

Lambda 演算与计算理论

由于图灵到处推销自己的理论，把不好的东西说成是好的，把别人发明的机器硬往自己的理论上面靠，说他们受到了图灵机的“启发”，以至于很多人被蛊惑，以为它比起 lambda 演算确实有优势。再加上很多人为了自己的利益而以讹传讹，充当传教士，这就是为什么图灵机现在被人们普遍接受作为计算模型。然而这并不能改变它丑陋和混淆的本质。图灵机的设计，其实是专门为了证明 Hilbert 的可判定性问题不可解决，它并不是一个用途广泛的计算模型。图灵机之所以被人接受，很大部分原因在于人的无知。很多人（包括很多所谓“理论计算机科学家”）根本没好好理解过 lambda 演算，他们望文生义，以为图灵机是“物理的”，实际可用的“机器”，而 lambda 演算只是一个理论模型。

事实恰恰相反：lambda 演算其实非常的实用，它的本质跟电子线路没什么两样。几乎所有现实可用的程序语言，其中的语义全都可以用 lambda 演算来解释。而图灵机却没有很多现实的意义，用起来非常蹩脚，所以只能在计算理论中作为模型。另外一个更加鲜为人知的事实是：lambda 演算其实在计算理论方面也可以完全取代图灵机，它不但可以表达所有图灵机能表达的理论，而且能够更加简洁和精确地表达它们。

很多理论计算机科学家喜欢用图灵机，仿佛是因为用它作为模型，能让自己的理论显得高深莫测，晦涩难懂。普通的计算理论课本，往往用图灵机作为它的计算模型，使用苦逼的办法推导各种可计算性（computability）和复杂性（complexity）理论。特别是像 Michael Sipser 那本经典的计算理论教材，晦涩难懂，混淆不堪，有时候让我都怀疑作者自己有没有搞懂那些东西。

后来我发现，其实图灵机所能表达的理论，全都可以用更加简单的 lambda 演算（或者任何一种现在流行的程序语言）来表示。图灵机的每一个状态，不过对应了 lambda 演算（或者某种程序语言）里面的一个“AST 节点”，然而用 lambda 演算来表示那些计算理论，却可以比图灵机清晰和容易很多。在 Indiana 大学做计算理论课程助教的时候，我把这种思维方式悄悄地讲述给了上课的学生们，他们普遍表示我的这种思维方式更易理解，而且更加贴近实际的编程。

举一个很简单的例子。我可以用一行 lambda 演算表达式，来显示 Hilbert 的“可判定性问题”是无解的：

```
Halting(λm.not(Halting(m,m)), λm.not(Halting(m,m)))
```

完整的证明不到一页纸，请看我的另外一篇文章（英文）。这也就是图灵在他的论文里，折腾了十多页纸证明的东西。

我曾经以为自己是唯一知道这个秘密的人，直到有一天我把这个秘密告诉了我的博士导师 Amr Sabry。他对我说：“哈哈！其实我早就知道这个，你可以参考一下 Neil Jones写的一本书，叫做《Computability and Complexity: From a Programming Perspective》。这本书现在已经可以免费下载。

此书作者用一种很简单的程序语言，阐述了一般人用图灵机来描述的那些理论（可计算性理论，复杂性理论）。他发现用程序语言来描述计算理论，不但简单直接，清晰明了，而且在某些方面可以更加精确地描述图灵机无法描述的定理。得到这本书，让我觉得如获至宝，原来世界上有跟我看法如此相似的人。

在一次会议上，我有幸地遇到了 Neil Jones，跟他切磋思想。当提到这本书的模型与图灵理论的关系，老教授谦虚地说：“图灵的模型还是有它的价值的……” 然而到最后，他也没能说清楚这价值何在。我心里很清楚，他只是为了避免引起宗教冲突，或者避免显得狂妄自大，而委婉其词。眼前的这位教授，虽然从来没有得过图灵奖，很少有人听说过他的名字，然而他对于计算本质的理解，却比图灵本人还要高出很多。

总的说来，图灵机也不是一文不值，然而由于 lambda 演算可以更加清晰地解释图灵机能表示的所有理论，图灵机的价值相对来说几乎为零。Church 在 1937 年给图灵论文写的 Review 指出，图灵机的优势，在于它可以让不懂很多数学，不理解 lambda 演算之类理论的人也可以看得懂。我怎么觉得图灵机对于不懂很多数学的人，理解起来其实更加痛苦呢？而且就算它真的对“外行”或者“笨人”的理解有好处，这价值貌似也不大吧？:P

电子计算机

很多“理论计算机科学家”喜欢说，大家现在用的计算机，只不过是一个“Universal Turing Machine”。就算你根本不知道图灵是谁，自己辛苦设计出一个机器或者语言，他们总喜欢说：“是图灵启发了你，因为你那东西是跟图灵机等价的，是图灵完备的……”

那么现在让我们来看看，图灵本人和他的理论，真正对电子计算机的发展起过多大的作用吧。如果一个人对一个行业起过重大的作用，那我们可以说“没有他不行”。然而事实却是，即使没有图灵，计算机技术会照样像今天一样发展，丝毫不会受到影响。看一看历史，你也许会惊讶的发现，图灵的理论不但没能启发任何计算机的设计，而且图灵亲自设计的唯一一个计算机（ACE），最后也以悲惨的失败告终。

什么是 Universal Turing Machine（UTM）

ACE 失败的一个重要原因，是因为图灵过度的看重他自己发明的 Universal Turing Machine（UTM）。所以我想首先来解密一下，这个被很多人吹得神乎其神的，似乎什么都可以往上面扯的 UTM，到底是什么东西。

说白了，UTM 就是一个解释器，就像 Python 或者 JavaScript 的解释器一样。计算机的处理器（CPU）也是一个解释器，它是用来解释机器指令的。那这样说来，任何可编程，具有指令集的机器都是 UTM 了，所以图灵的理论启发了所有这些机器？你尽管跟我扯吧 :)

你应该知道，在图灵的 UTM 出现以前，Church 的 lambda 演算里面早就有解释器的概念了，所以 UTM 不是什么新东西，而且它比起 lambda 演算的解释器，真是丑陋又复杂。而 Church 其实也不是第一个提出解释器这概念的人，像这类通用的概念，已经很难追溯是谁“发明”的了。也许并不是某一个人发明了它，而是历史上的很多人。

解释器这个概念的涵义实在是包罗万象，几乎无处不在。只要是“可编程”的机器，它本质上必然包含一个解释器。一个工程师在不知道解释器这概念的情况下，照样很有可能“不小心”设计出一个可编程的机器，所以如果你把这些全都归结成图灵或者 Church 的功劳，就太牵强了。

图灵与 ACE 的故事

事实上，最早的电子计算机，并不是图灵设计的，而是电子工程师跟其他一些数学家合作的结果。根据老一辈工程师的叙述，图灵的工作和理论，对于现实的电子计算机设计，几乎没有任何的正面作用。很多工程师其实根本不知道图灵是谁，图灵机是什么。他们只是根据实际的需求，设计和制造了那些电路。这就是为什么我们今天看到的电子计算机，跟图灵机或者图灵的其他理论几乎完全不搭边。

世界上最早的两台电子计算机，ENIAC 和 EDVAC，都是美国人设计制造的。其中 EDVAC 的设计报告，是冯诺依曼（von Neumann）参与并签署的。提到 EDVAC 的设计，《图灵传》有一段有趣的介绍，它基本是这样说的：“冯诺依曼在 Princeton 的时候，很了解图灵开天辟地的发明—UTM。UTM 只有一根纸带，而 EDVAC 把指令和数据放在同一个存储空间，所以 EDVAC 的设计肯定是受了 UTM 的启发。然而 EDVAC 的设计报告，却只字不提图灵和 UTM 的名字，更没有引用图灵划时代的论文《Computable Numbers》……”

这其实是在含沙射影的说，冯诺依曼和 EDVAC 团队抄袭了图灵的研究成果。照这种歪理，我洗衣服的时候，袜子和内裤放在同一个桶里洗，也是受了图灵的启发了，就因为 UTM 只有一条纸带？这世界上的事物，还有什么不是受了 UTM 启发的？这让我想起某些全靠打专利官司赚钱的公司（patent troll）…… 冯诺依曼作为一代数学大师，比 UTM 重大的研究成果多得是了，他会在乎抄袭图灵的东西吗？其实人家恐怕是根本没把图灵和他的论文当回事。而且其他人（比如 Church）早就有跟 UTM 等价的想法，而且还更好，更简单。之前抢了 Church 的风头，现在居然欺到冯诺依曼头上来了。哎，真受不了这种一辈子只想出过一个点子的人 ;)

所以听说美国人造出了 EDVAC，图灵开始各种羡慕嫉妒恨，感叹自己英才无用武之地。终于有一天，他的机会来了。在 EDVAC 诞生几个月之后，英国国家物理实验室（NPL）联系了图灵。他们想赶上美国的计算机技术发展，所以想招募图灵，让他帮忙山寨一个 EDVAC 的“英国特色版本”。图灵设计的机器叫做 ACE（Automatic Computing Engine）。最初，图灵给 NPL 一个很宏伟的蓝图：ACE 可以如此的强大，以至于整个英国只需要这样一台计算机就够了，我们可以把它叫做“英国国家计算机”…… 然而再大的口号，也难逃脱现实的检验，ACE 项目最终以失败告终。

《图灵传》把 ACE 失败的责任，推托到 NPL 和其它人的“近视”和“官僚”，然而 ACE 失败的主要责任，其实在于图灵自己：他没有设计一台现实的计算机的基本技能，却设立高大空的目标。图灵的设计跟当时（包括现在的）所有实用的计算机都有巨大的差别。不出你所料，他最初的设计思路，是根据自己的 UTM，不过从中去掉了一些不实际的设计，比如用一根纸带来存储数据。这一点改进貌似做对了，可是呢，他又加入了一些让工程师们无语的设计，美其名曰“极简设计”（minimalism）。比如，ACE 的硬件只提供 AND, OR, NOT 之类的逻辑运算作为“基本操作”，其它的算数操作，包括加减乘除，全部用代码来实现。图灵大师啊，你知不知道有一种重要的指标，叫做“效率”？

这还不算…… 后来他更加异想天开，终于扯上了“思考机器”（thinking machines）—他想让 ACE 成为可以像人一样思考的机器，还想让这机器能够自己写自己的代码。按照图灵的原话：“在 ACE 的工作中，我对人脑建模的兴趣，比实际的计算应用更感兴趣。” 他显然已经把 ACE 当成了自己一个人的玩具，而不再是解决人们实际需求的工具。只要有人反对这想法，他就会嘲笑说，你是怕我的机器太聪明了，抢了你的饭碗吧？其实图灵对于实际的人脑工作原理所知甚少，基本处于初中生理卫生课本水平，然而他总喜欢对人说，人脑不过就是一个 UTM。看吧，它有输入，输出，状态转换，就跟 UTM 一样…… 所谓“图灵测试”（Turing Test），就是那时候提出来的。当然了，因为他扯到了“thinking machine”，就有后人把他称为人工智能（AI）的鼻祖。其实呢，图灵测试根本就不能说明一个机器具有了人的智能，它只是在测试一些肤浅的表象。后来，“thinking machines”成为了一种通用的幌子，用于筹集大笔科研经费，最后全都血本无归。

图灵设计了这机器，NPL 当时却没有能力制造它。于是他们求助于另外两位实现过计算机的工程师：F. C. Williams 和 Maurice Wilkes（后来 EDSAC 计算机的设计者），请他们帮忙实现图灵的设计。可想而知，Williams 和 Wilkes 都表示不喜欢 ACE 的设计，而且指出图灵的性格与自己的研究风格不匹配，不愿跟他合作，所以双双拒绝了 NPL 的邀请。最后，NPL 新成立了一个电子部门，ACE 的工程终于可以开始。然而，根据资深工程师们的讨论，觉得图灵提出的制造一个“电子人脑”和“智能机器”，并不是实际可行，或者在短期之内能派上用场的项目，所以决定做一些实际点的事情。图灵对此非常恼火，各种抱怨，说别人官僚啊，近视啊，没想象力啊之类的，然后开始公开的抵制 NPL 的决定。

最后工程师们和管理层都受不了他了，鉴于他名声在外，又不好意思开掉他，只好提出一个破天荒的提议：由 NPL

资助，让图灵回到剑桥大学去度年假（sabbatical），做一些纯数学的研究。于是 ACE 在图灵不在的情况下，终于可以开工了…… 1950 年，ACE 运行了它的第一个程序。然而工程师们实现的 ACE，完全偏离了图灵的设计，以至于实际的机器和图灵的设计之间，几乎没有任何相似性。一年之后，图灵还想回到NPL，继续影响 ACE 的设计，然而 NPL 的领导们却建议他继续留在大学里做纯理论的研究，并且让曼彻斯特大学给他一个职位。最后图灵接受了这个建议，这下大家伙儿都松了一口气…… :P

图灵设计的唯一一个计算机 ACE，终究以图灵完全退出整个项目而告终。今天回头看来，如果当时图灵留下来了，NPL 真的按照图灵的意思来做，ACE 恐怕直到今天都造不出来。由于图灵不切实际的设计和高傲的性格，NPL 失去了最优秀的人的帮助。1949年，Maurice Wilkes 按照 EDVAC 的思路，成功制造了 EDSAC，速度是 ACE 的两倍以上，而且更加实用。

如果对 ACE 和其它早期计算机感兴趣，你可以参考一下更详细的资料。你也可以看一看《图灵传》，虽然它观点荒唐，对图灵各种偏袒，然而图灵和其他人的通信，基本的史实，他应该不好意思篡改。

总结

我说这些是为了什么呢？我当然不是想否认图灵所做出的贡献。像许多的计算机工作者一样，他的工作当然是有意义的。然而那种意义并不像很多人所吹嘘的那么伟大，它们甚至不包含很多的创新。

我觉得很多后人给图灵带上的光环，掩盖了太多其它值得我们学习和尊敬的人，给人们对于计算机科学的概念造成了误导。计算机科学不是图灵一个人造出来的，图灵并不是计算机科学的鼻祖，他甚至不是在破解 Enigma 密码和电子计算机诞生过程中起最重要作用的人。

许许多多的计算机科学家和电子工程师们，是他们造就了今天的计算科学。他们的聪明才智和贡献，不应该被图灵的光环所掩盖，他们应该受到像跟图灵一样的尊敬。希望大家不要再神化图灵，不要再神化任何人。不要因为膜拜某些人，而失去向另一些人学习的机会。