可计算性
计算复杂性
计算模型的相似性
时间和空间的对偶性

计算的相似性与对偶性原理——谈谈理论计算机科学

THE THEORY OF ANALOGY & DUALITY OF COMPUTATION

洪加威

电子计算机的发明和发展引起了一场伟大的技术革命，其规模之大，范围之广，变革之深刻，都是人类历史上少有的。

比如说，计算机刚发明的时候，人们多半认为它只是用来作计算用的，但是，今天我们已经把它用在文件编写、资料整理、生产控制、密码通信，甚至绘图、设计、制表、排字、订票、游戏等等看起来极不相同的领域中，许多问题和计算的传统概念简直是风马牛不相及。

于是就产生了一个问题：计算机到底能作些什么？比如说，会不会写小说？会不会制定交通规则？会不会证明数学定理？要回答这类问题，首先要回答一个更理论性的问题，这就是：什么是计算？

理论计算机科学的任务之一，就是研究与此有关的问题。

理论计算机科学产生于30年代。那时候，人类刚刚有了现代计算机的想法，但是第一架计算机还没有开始研制，不过最有远见的人们已经想到计算机可能带来的理论问题。于是，他们就开始研究哪些问题是“可计算的”。所以，这个时期的理论计算机科学主要是研究可计算性理论，或者叫能行性理论。

后来，计算机造出来了，广泛应用起来了，人们又发现了另一个重要的问题：有的问题，虽然理论上是可计算的，但是实际上却由于所需要的时间和空间（指需要暂时存储起来，下面再利用的中间结果所占用的存储器）超过了实际可能，所以无法进行计算。可计算性理论不能回答这个问题。回答这个问题的是理论计算机科学的另外一个分支——计算复杂性理论。这是理论计算机科学最新、最活跃的分支。可以说，从可计算性理论到计算复杂性理论，这是理论计算机科学发展的一个重要方面。本文就是打算介绍这一方面的一些成果。

什么是可计算的？

请看下列问题：

A. $35+27=?$

B. $35\times27=?$

C. 解方程组：

$$\begin{aligned}3x+5y&=8\\7x-4y&=3\end{aligned}$$

D. 判断“三角形的三中线必相交于一点”是不是一条定理。

E. 解某个象棋残局，判断先走者（红方）是不是有办法赢。

F. 判断一个计算机的程序如果运行起来会不会有终于算完的一天。

这些问题可以用计算机来计算吗？换句话说，能不能把这些问题输入到计算机里去，让计算机把答案输出出来呢？

要计算机工作，首先要给它编写一个程序，告诉它先作什么，后作什么，遇到什么情况该作什么等等。所以，能不能计算的问题就归结为能不能编出解题程序的问题。

但是，这个问题的提法有些毛病。拿问题A来说吧，这个问题的答案是62。那么，如果编一个程序，不让计算机进行任何计算，只让它输出62这个数，这算不算计算机解答了这个问题呢？又比如说问题D，如果编一个程序，什么也不干，只输出一个“是”（用1代表）就停机，算不算计算机会解这个几何题呢？

上面的每一个问题都可以这样作：我先找到答案，

洪加威（J. W. Hong）理论计算机科学家，罗彻斯特大学（Rochester U.）和卡内基·梅隆大学（Carnegie-Mellon U.）中国访问教授，北京市计算中心研究员。他提出的相似性对偶性原理，是理论计算机科学的一个前沿。

再编一个程序，让计算机把这个答案输出来。这当然不能算是计算机真的在解这个问题，这是假的。可见，这些问题“能不能用计算机来计算”是个真假难分的问题。

其实，当我们考虑计算机会不会解第一个问题的时候，我们是在问“计算机会不会求两个整数的和”。换句话说，我们并不是讨论某一个特定的问题，而是某一类问题。我们需要的程序，也不是只能输出那个62的程序，而是这样的一个程序：不论我们把两个什么样的整数输入到计算机里，计算机都能输出和数来。这样的程序一旦编制出来，我们就可以说，计算机会作加法，也就是说加法是可计算的了。

可见，上面的那几个问题应该重新提成：

A. 求两个整数的和。

B. 求两个整数的乘积。

C. 解线性方程组。

D. 判定初等几何定理。

E. 解象棋残局，回答谁胜谁负。

F. 判定任何给定的程序会不会终于停机。

当然，这些问题的类可以提得宽一些或者窄一些，但重要的是，他们都不是一个个别的问题，而是一类问题（而且，这一类问题有无限多，不能一一列举）。这样，我们就不可能把答案都事先算出来，然后编那种假的程序，而必须编出真能解这一类问题的程序来。

比如说我们编了一个作乘法的程序，那么，如果把12和3输入到计算机里，计算机就应输出36，如果把20和30输入到计算机里，计算机就应输出600，如此等等。总之，有了解某一类问题的程序以后，再要解这一类问题中的某个问题，只用把这个问题（指其中的数据、系数、图形、规格等等）输入到计算机里去，计算机就可以输出所要的答案。

这样，可计算性的概念，总算比较清楚一些了。

但是，还有不清楚的地方。有各种各样的计算机，将来，还会不断有新型的计算机出现，那么，我们说可计算性时指的是哪一种计算机呢？

我们当然不能把每一个计算机分别讨论一遍。我们也无法知道将来的人类会作出什么样的计算机来。所以，理论计算机科学必须把计算机模式化，就是设计出一些作为理论研究的工具的计算机，讨论这种计算机能解什么问题。这种计算机能表现各种计算机的计算过程，但是应该略去一切次要的细节（怎样输入，怎样输出，用磁带还是磁盘作外部存储器等等），这种计算机是纸上谈兵的计算机，通常把它们叫做计算模型，图灵机（Turing machine）就是一个这样的计算模型。

图灵机（图1）由一个分成许多格子的带子（简称带）和一个可以在带上左右移动的读写头（简称头）组成。图灵机的工作分成许多步骤（简称步）。在每一步里，图灵机可以作下列工作之一：使头在带上左右移动一格；在头所面对的格子里写上指定的符号；

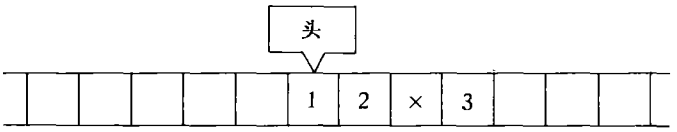

图1　图灵机

读头所面对的格子里的符号，并根据这个符号是什么来决定作什么；停止计算。为图灵机编写的程序就由许多这样的步骤组成。图灵机没有输入输出装置，计算开始的时候，假定要解的问题已经写在带上，计算停止的时候，带上的符号就是答案。

图灵机只是种种可能的计算模型之一。现代理论计算机科学中用到的主要计算模型有十二种之多。但是，有一个有名的论题说明了图灵机的价值，这个论题是：

却其—图灵（Church—Turing）论题　任何有足够计算能力的计算模型都等价于一个图灵机。

换句话说，一类问题如果能用某种计算模型解出，也一定能用图灵机来解，反之，一类问题如果不能用图灵机解出，任何别的计算模型也无可奈何。

于是，一类问题是不是可计算的，就是一个客观的事实，並不依赖于计算的模型。所以却其—图灵论题为可计算理论奠定了基础。

这个论题不能用数学方法来证明，因为没有办法精确定义什么是“有足够计算能力的计算模型”。但是，图灵机以后的各种计算模型确实都和图灵机等价，无一例外。由此可见图灵的工作的重要性。所以，现在大家纪念他，为计算机科学设立了“图灵奖”。

到此为止，我们已经明确了什么叫做可计算性了。但是，到底哪些问题是可计算的，哪些不是，还得一个一个地研究。在初等数学中我们就知道上面的A，B，C这些问题都是可计算的。但是更有趣的事实应该是：

定理　初等几何问题的证明是可计算的。

换句话说，可以给图灵机编一个程序，只要你把想要证明的定理事先写在带上，那么，当图灵机停止计算的时候，带上就写着这个定理的证明。这简直是懒学生的福音了。可惜，现在已知的任何这种程序，因为计算的步骤太长，无法在一段合理的时间里证明出有价值的定理来，所以还没有实用意义。

到现在为止，已经知道的可计算的问题非常之多。有许多还编出了可以实际使用的程序。正是因为这个缘故，计算机的使用才能有今天的规模。但是，计算机决不是万能的。很可能大多数问题都是不可计算的，不过，要证明一个问题不可计算往往非常困难，所以现在我们还很难举出许多例子来。

下面是一个不可计算的问题的例子：

设给了两组字：

A＝{ a b，b a c a，c c b b a，a }

B＝{ a b a b，a c a a b }

问有没有办法把A中的某些字（可以重复使用）排成一串，把B中的某些字也排成一串，并使这两串字完全一样呢？对于上面的例子，这是可能的，比如：

a b a b a c a a b＝a b a b a c a a b

能不能编一个程序，对随便给的两组字来判定这个问题（即回答“能”或“不能”）呢？

回答是：

定理 这个问题是不可计算的。

用不着再费脑筋去编这种程序了，永远编不出来的。

上面提到的问题F也是不可计算的（不可判定的）。

定理 停机问题不可判定。

这个定理的意思并不是说我们根本不知道哪个程序会停机，哪个程序不会停机，而是说不存在一个统一的机械方法（即不存在一个统一的程序）去做这件事。

什么是计算的复杂性？

先讲一个童话。

很久很久以前，有一个年轻的国王，名叫艾述。他非常喜欢数学，所以把当时最大的数学家孔唤石请来当宰相。

他的邻国有一位聪明美丽的公主，名叫秋碧贞楠。艾述国王爱上了她，便亲自登门去求婚。

公主说：“‘求’是比‘证’要难的。你如果向我求婚，请你先求出48770428644836899的一个真因子，一天之内交卷。”艾述国王一听，心中暗喜，他想，我从2开始，一个一个地试，看看能不能除尽这个数，还怕找不到这个真因子吗？

艾述国王果真是精于计算，他一秒钟就算出一个除法。从早晨开始，到吃午饭的时候，已经算到一万多，可是没有一个数能除尽这个怪数。他胡乱吃了几口午饭，又拿起小黑板，急急忙忙算下去。到了晚上，试过三万个数了，还是不行。公主说，你既然求不出来，那么请你验证一下，223092871是它的一个真因子。国王如获大赦，在一秒钟之内就做了一个除法，证明了这个数恰恰能除尽公主给的那个数。公主于是说，你求婚不成，将来做我的证婚人吧。但是我可以给你一个机会再求一次。

国王急急回国，召见宰相孔唤石。这位大数学家说：“这个问题我也没有什么好办法，不过你可以这样试一下：你把全国分成10个军，每军下分10个师，下面再分旅、团、营、连、排、班，每班10个人。公主的数目不超过17位，如果这个数可以分解成两个因子的乘积，那个较小的因子一定在9位以下。你叫每个老百姓记住自己是哪一军哪一师，……，哪一班的。公主给了一个数目以后，你把这个数通报全国，叫每个老百姓用自己的番号去除这个数。例如第一军二师三旅四团五营六连七排八班的第九个士兵，应该用123456789去除公主的数。谁除尽了赶快层层上报，赏黄金300两。”

于是国王发动群众，再度求婚，终于成功。后来的事就象一切故事那样——我不讲你也知道。

故事讲完了。这个故事中提出来的问题，就是求一个整数的因子的问题，当然是可计算的。而且，艾述国王开始想的办法——从2开始，一个一个地试验的办法，就可以写成一个图灵机的程序。这个办法的失败，并不是由于计算方法不对，而是因为按这个方法进行计算所需要的时间太长。

可见，讨论一个问题能不能用计算机求解，不能光讨论可计算性，还要考虑需要多长时间。于是，我们就进入了计算复杂性理论的领域了。

计算复杂性理论中最容易明白的概念就是计算时间。其实，除了时间以外还有别的问题。在这个故事中，孔唤石建议的办法，就可以节约时间。

如果我们用黑板来计算除法，按照第一个办法，只用一块黑板就够了，因为前面的计算对后面并没有用处，可以擦去；按照第二个办法来做，就得每人一块小黑板，因为所有的计算是同时进行的。这里，黑板相当于计算机的存储器。所以，第一个办法复杂在时间方面，第二个办法复杂在空间方面。

在计算机科学中，我们把第一种方法叫序列的计算方法，第二种叫并行的计算方法。在这个故事里，很容易看出，并行计算节省时间，是因为很多计算一齐作，总的计算量并没有减少，是用空间换取了时间。

艾述是个大国的国王，所以他可以这样做，换个普通的小伙子，想找秋碧贞楠公主当妻子谈何容易！不过他可以用下面的办法去碰碰运气：瞎猜一个数，用它去除公主的数。这种可能性当然很小了。不过万一碰着了呢？反正试着除一下也不怎么费事。

这种普通小伙子的猜测—验证的方法，叫做非决定型（non-deterministic）的计算方法。而通常的计算方法就叫做决定型（deterministic）的方法。容易看出，这种非决定型的计算方法，所用的计算量是很省的。而结果就难说了，要看各人的运气了。有什么办法呢？这是没有办法的办法呀！

现在很多人猜想秋碧贞楠公主的看法是对的：求解一定比验证难（或者用数学的行话说，叫做 NP≠P）。这个猜想是一个极重要的问题，到现在还没有证明。

上面说的时间、空间等等在计算机科学中统称资源。要解一类问题至少需要多少资源，叫这一类问题的计算复杂性（简称复杂性），当然，一般说来，同一类问题中的问题有大有小，在上面的故事中，如果把那个怪数换成一个很小的数，那么，整个的故事就完全改观了。所以，计算复杂性并不是一个数，而是问

题的大小（如未知数的个数，棋盘的大小，等等）的函数。所以，我们常说某一类问题是“多项式复杂性的”或“指数复杂性的”等等，就是说这类问题的复杂性是问题大小的多项式函数或指数函数。

决定型、非决定型等等叫做计算的类型。复杂性要分别类型进行讨论。除了决定型、非决定型以外，在计算机科学中还有许多其他的计算类型，如交错型（alternating）、随机型（random）等等。

计算模型的相似性

一类问题的计算复杂性是否与计算模型有关系呢？会不会有些类的问题在图灵机上计算，所需资源很多，如果换一种计算模型做，所需资源却比较少（或者相反）呢？这叫做计算模型的相似性问题，是计算复杂性理论的基本问题，相当于可计算性理论中的却其一图灵论题。

但是，各种计算模型在形式上极不相同，所以它们用的资源也极不相同。

拿图灵机说吧。在计算机科学中常常要用多带图灵机。这是一种有许多条带子的图灵机，每一条带都有一个读写头。但是，只有一个程序，机器是一步一步地工作的，是一个序列的计算模型，不能作並行计算。

拿多带图灵机和並行的计算模型相比，什么是相当于並行的计算时间呢？原来，图灵机在计算时，头是左右移动的，如果在一段时间里，所有的头都没有改变移动的方向，这一段时间就叫一个周相(phase)，计算过程中的周相总数叫做巡回（reversal）。巡回也是一种重要的资源，它就相当于並行计算时间。这种对应关系就不是直接的对应而是比较间接的了，要研究它们的关系就会遇到困难。

还有更间接的例子。有一种计算模型叫齐一线路(uniform circuits，图2)。这是用与门、或门、非门做成的逻辑线路，其中最下面的是第零层，它的上面是第1层，再上面是第2层等等，每一层的输入都连接到下面各层中的某个门。整个的线路如何安排，要看想解什么问题而决定。线路安排好以后，把问题编成二进制的码，从第零层输入到线路里来，在最上面一层就得到答案。在一个齐一线路中，总的层数叫线路的深度，通过每一层的线的数目的最大值叫线路的宽度，整个线路中门的总数叫线路的大小。齐一线路是一种並行的计算模型。

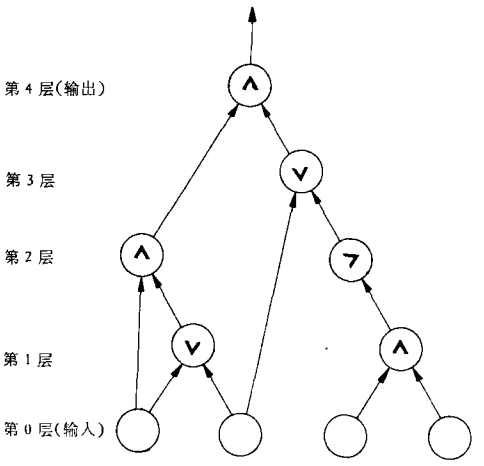

图2　齐一线路计算模型

怎样把齐一线路和多带图灵机比较呢？原来，齐一线路的深度、宽度和大小分别相当于多带图灵机的巡回、空间和时间。更准确的说法如下：

用R，S，T代表多带图灵机的巡回、空间和时间；用D，W，C代表齐一线路的深度、宽度和大小；用n^*表示n的某个多项式。可以证明：

（1）对任何多带图灵机（的程序）都可以找到一个齐一线路，使得对任何的输入而言，它们的计算结果相同，而且

$$D \leqq R^*$$
$$W \leqq S^*$$
$$C \leqq T^*$$

（2）对任何齐一线路都可以找到一个多带图灵机（的程序），使得对任何的输入而言，它们的计算结果相同，而且

$$R \leqq D^*$$
$$S \leqq W^*$$
$$T \leqq C^*$$

也就是说，如果有一类问题可以用多带图灵机来解，那末也可以用齐一线路来解，而且它的深度、宽度和大小与多带图灵机的巡回、空间和时间相差不多（在研究计算复杂性理论的时候，我们把相差一个多项式的复杂性认为是一类的），反过来也一样。这就是说，这两个计算模型在研究计算复杂性的问题时，并无本质的差别，或者说他们是相似的，可互相模拟。

相似性定理

从上节的讨论可以看出，要讨论相似性的问题，就要揭露各种计算模型的种种不同的资源的本质。比如说，在並行计算模型中，相当于序列计算模型中的计算时间的东西是总计算量。这个看法在前面的故事里已经可以看出来了。但是，在序列的计算模型中相当于並行计算时间的是什么呢？前面曾说，对于多带

---

*　这是一个简略的说法，意思是这样的一个算法，其复杂性是多项式函数。下同。

图灵机来说，相当于并行计算时间的是巡回，这是为什么呢？下面我们就来谈谈这个问题。

在多带图灵机的每一个周相中，读写头不改变移动方向，所以在每一个周相中写下来的东西，在同一个周相中是不会被读到的。这是周相最本质的地方。我们用这个观点来考查各种序列模型，就可以找到什么是那个计算模型的周相。周相的总数就叫巡回，也就是这个计算模型中相当于并行计算时间的东西。

为了弄明白这个问题，我们还要说到那个故事。在艾述国王的第一个算法中，如果他这样做：先找许多黑板，在不同的黑板上用不同的数去除公主给的数。在每一块黑板上写上一个算式，然后再在每一块黑板上写上商数的第一位，如此等等，总之在一块黑板上写上一个数，就转向下一块黑板，等到每一块黑板都这样作了，再回到第一块黑板写下一个数。他这样做，除了要跑来跑去以外，和原来的作法并没有什么不同。这时我们就看出：他从第一块黑板跑到最后一块黑板，在这一段时间里分别在不同的算式里各写了一个数，所以，这些数在这一段时间里是用不着再去读的，这一段时间就是一个周相。一块黑板上最多会写上多少数字，就是巡回。其实，这不正是那个故事中第二种（并行的）计算方法要用的时间吗！

再看齐一线路。因为每一层的输入都连接到下面各层，所以，每个门的输出（相当于写出的东西）就不会在同一层中用到。这样，每一层就是一个周相，层数（也就是深度）就相当于并行计算时间了。

序列计算时间和并行计算时间的概念本来是和计算模型有关的概念，现在我们把他们分别变成了总计算量和巡回，这就是不依赖于计算模型的概念了。有了这种统一的概念，就可以开始研究相似性问题。研究的细节非常复杂，远远超出了本文的限度，但是，无论如何，我们终于证明了：

相似性定理 现有主要的十二种计算模型都是相似的。

以前我们说过，各种计算模型能解的问题是一样多的，现在，我们可以更进一步说，各种计算模型不但能解的问题一样多，而且解问题时用的资源也一样多。所以，不但可计算性的概念与所用的计算模型没有关系，计算复杂性的概念也与用什么模型没有关系。

我们证明了所有现有的十二种计算模型的相似性，就可以说：一类问题的计算复杂性是不依赖于计算模型的。这是计算复杂性理论的基础，可以说是却其一图灵论题在可计算性理论中的新发展。

广义相似性定理

相似性定理只解决了决定型模型的相似问题，非决定型模型呢？交错型模型呢？

在经典的定义下这个问题是不便解决的。这是因为在经典的定义中，计算类型是分别对不同的计算模型定义的，比如，怎样定义非决定型的图灵机与怎样定义非决定型的齐一线路是不相干的。这种定义没有暴露问题的本质，所以不够深刻。我们必须有一种新的定义，把各种不同的计算模型中计算类型的定义统一起来。这种定义抛弃了旧有的概念，又把旧有的概念看成自己的某种特殊情况，在一定的限制下的特殊情况。这和前一节中用巡回代替并行计算时间是类似的。

我们先看非决定型的图灵机。非决定型的图灵机可以看成在图灵机上写上了非决定型的程序，或者用“行话”来说，就是在程序的不同地方可以出现相同标号。计算到这里的时候，不知道应该走哪一条路，需要猜测。

那么，如果我们有一个解问题w的非决定型的程序。我们先请一位好运气的人把需要猜测的地方预先给我们作好提示，再把这些提示用一串符号g附在问题后面，写成w # g的样子（这里#是一个符号，表示从这里以后是提示）。然后，我们就可以编出一个决定性的程序来进行同样的计算。其实，这个程序完全和原来的非决定性程序一样，只是在需要猜测的时候，就去读后面的提示，并根据提示决定如何接着计算下去。

我们当然并不知道对于一个问题w需要写出什么样的提示g来，但是，一类问题如果能用非决定型的图灵机来解，这种提示就一定存在。如果所要解的一类问题是判定问题（即所要算的答案是零或1，前者称为机器拒绝输入，后者称为机器接受输入，对于计算模型的讨论常常只就这类问题进行），就可以把非决定型的图灵机定义成：

非决定型图灵机是这样一个机器，对任何输入w而言，如果存在一串符号g，使得决定型图灵机接受w # g，它就接受w，反过来，如果不存在这样的g，它就不接受w。

非决定型图灵机的这种定义脱离了图灵机的构造细节，抓住了本质，很容易推广为一个一般的概念：

设M是一个决定型的计算模型。和M相应的非决定型计算模型是这样一个计算模型NM，对任何输入w而言，如果存在一串符号g，使得M接受w # g，则NM就接受w，反过来，如果不存在这样的g，NM就不接受w。

我们找到了非决定型计算模型的一般概念。我们还可以类似地找到其他类型的计算模型的一般概念。这些概念都要用到一串符号g，只是意义不同，在定义中怎样起作用也有区别。比如在解象棋残局问题中用到的交错型计算模型中，g的每个符号可能代表双方的每一步走法，等等。因此，计算类型最一般的定义就是：

类型是一种原则，这种原则可以决定如何根据一个决定型的计算模型对不同的g是否接受w # g来确

定该类型的相应计算模型是否接受w。用数学的行话来说，类型是一个“泛函”。

为了把一般的定义和经典的定义联系起来，我们只需要对它做一些限制，有两种限制：

限制S（序列的限制，sequential restriction）：规定g的每一个符号只许用一次；

限制P（并行的限制，parallel restriction）：要求g的长度不能超过并行时间。

经典的定义要么相当于限制S（序列计算模型的情形），要么相当于限制P（并行计算模型的情形），要么相当于同时有限制S与限制P〔向量机（vector machines）计算模型的情形〕。

现在我们可以讨论各种类型的计算模型的相似性问题了。

前面已经说过，十二个现有的计算模型都相似，那是指决定型模型而言的。换句话说，这十二个决定型的计算模型都相似于多带图灵机。那么，对非决定型或者别的类型又如何呢？对这个问题的回答也是肯定的：对任何类型T，十二个计算模型都相似于类型T的多带图灵机。不但如此，我们还可以证明，对任何类型T和任何限制R，十二个计算模型都相似于在限制R下的类型T的多带图灵机。

到了这一步，我们已经可以提出下面的原理了：

相似性原理：在限制R下的类型T的任何理想计算模型都相似于在限制R下的类型T的多带图灵机。

不用说，这当然是从许多方面加强了却其一图灵论题。

各种计算模型本来是极不相同的东西，但我们有了相似性定理，在讨论一类问题的复杂性的时候，只对一个计算模型讨论就够了。常常有这样的情况：一类问题的复杂性对某个计算模型讨论很困难，对另一个计算模型讨论却很容易。今后，遇到这样的情况，我们只用选一个适当的模型来讨论，所得的结果自动地对别的模型也有效。换句话说，我们只须证明一个较容易的定理，就可以自动地得出一大批定理，其中有一些甚至是很难证明的。正是因为相似性定理有这种化繁为简的妙用，我们才能轻而易举地把计算复杂性分类理论概括成下节中的诸元定理。

时间和空间的对偶性

通过对于计算模型的深入研究，我们发现了极为重要的时间空间对偶性（Duality）。这就是我们要谈的诸元定理。例如，我们有：

对称定理 对任何一个计算模型，任何一个计算类型和任何一类计算问题，

（1）如果存在一个多项式空间的算法*，就有一个多项式并行时间的算法；

（2）如果存在一个多项式并行时间的算法，就有一个多项式空间的计算。

也就是说，如果一个计算能在一个小的空间里完成，就一定能用短的时间并行完成，反过来，如果一个计算能用短的时间并行完成，就一定能在一个小的空间里完成。例如，前述的故事里的问题，按第一个算法来做（只要时间足够），就可以只用一块黑板完成，这是在一个小的空间里计算；如果按第二个算法做（只要黑板和人数足够），可以用短的时间并行完成。这就说明，在计算问题上，时间（指并行时间）和空间这两种不同性质的资源是可以互相转化的。这个事实，粗看起来，很容易想到，但是在经典的定义下，这个定理一般地并不成立。这说明了经典定义的局限性。

我们还发现，并行时间和空间不但是可以互相转化的，而且是互相制约的：

限制定理 对任何一个计算模型，任何一个计算类型和任何一类计算问题，

（1）并行时间不能超过空间的某个多项式的指数函数；

（2）空间不能超过并行时间的某个多项式的指数函数。

对称定理和限制定理都是以对偶形式出现的，如果我们在（1）中把“空间”读为“并行时间”，把“并行时间”读为“空间”，就是（2）。这种以对偶形式出现的定理还有很多，举例如下：

第一非决定型定理 对任何非决定型的计算模型和任何问题，

（1）如果有一个多项式空间的算法，就有一个多项式指数时间的算法，反之亦然；

（2）如果有一个多项式并行时间的算法，就有一个多项式指数时间的算法，反之亦然。

第一交错型定理 对任何交错型的计算模型和任何问题，

（1）如果有一个多项式空间的算法，就有一个多项式指数时间的算法，反之亦然；

（2）如果有一个多项式并行时间的算法，就有一个多项式指数时间的算法，反之亦然。

还有三个定理，我们就不详述了。

根据这些事实，我们就可以提出：

对偶性原理 如果我们有一个关于空间和（或）并行时间的定理，把空间和并行时间交换位置，就得到另一个定理。

时间、空间和并行时间是计算复杂性的三个要素，时间是总计算量，它相当于空间和并行时间的乘积，自己跟自己对偶，空间和并行时间互相对偶。计算复杂性理论的最新发展通过严谨而艰深的研究，给我们描绘了这样一幅最自然不过的图景。

最后，作者愿在此对马希文教授表示深切的感谢。由于他的许多建议，使本文能以比较通俗的形式与读者见面。