昆明师范学院
国庆卅周年学术报告集

自然科学

国庆三十周年学术报告汇集

（自然科学专辑）

目　　录

人脑、计算机和数学

刘声烈

内 容 提 要

一、麦克卡洛克和比脱斯关于人脑的一个模型 ——模块网络
二、另一个数学模型 ——有限自动机
三、自动机与数字电子计算机
四、图灵机
五、通用图灵机——一个万能程序
六、停机问题——一个不能判定的问题

一、麦克卡洛克和比脱斯关于人脑的一个模型——模块网络

大脑究竟是怎样发生作用的呢?我们能否设计一种与大脑相仿佛的机器?在历史上，许多科学家都提出了这样的问题。从过去几个世纪的文献来看，思考这些问题的方法，在任何一个时期都反映出当时人类所用机器的特征。笛卡尔（*Descartes*）在《论人类》这部巨著中，认为大脑与水钟、喷水泉、以及十七世纪广泛应用的各种机械设备有若干相似之点，认为神经细胞是通过微小的机械运动来传递信号的。在本世纪初叶，当自动电话系统诞生的时候，人们常常把神经系统比作通过自动开关设备来处理大量感受数据和执行数据的大型自动电话交换台。现在，人们则习惯于把大脑与大型电子计算机相比拟。基于神经解剖学和神经生理学研究的一些结果，在本世纪四十年代，麦克卡洛克（*W.S.McCulloch*）和比脱斯（*W·Pitts*）提出了一个神经系统的抽象模型——模块网络。

任何有机体都是通过自已的感受器官接受刺激，通过自已的执行器官进行动作。我们可以将人的神经系统视为一三阶段系统，如图 1 所示。

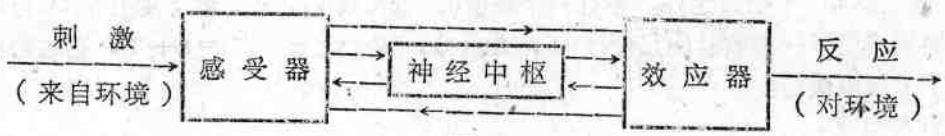


图 1

输入信息进入称做感受器的组织，输入能量即变为电信号。（所谓感受器，除了五官等感受器外，还包括感觉肌肉张力、身体平衡、血压、血糖值等状态的体内感受器。）变换后的电信号，通过构成传输通道系列的神经细胞键，到达中枢神经系统。在这里，经过复杂的神经网络，进行信息处理，通过所选定的适当输出系统，将输出信号送到末稍效应器。在效应器内通过电化学的变换，使肌纤维收缩、弛缓或神经分泌。

如图2所示，神经细胞是由树状突起、轴索及细胞体等构成。经由下述通道，信号从神经细胞A传达到神经细胞B：在神经细胞A的轴索末端产生电位变化，末端的小细胞就向神经细胞B附近的细胞膜移动，将内部的传送物质（乙酰胆碱、γ—氨基丁酸等）释放到A、B神经细胞膜之间的间隙内。传送物质传达到神经细胞B的树状突起膜上后，就改变了膜离子的电阻，使膜内产生电位，此电位向B神经细胞体方向传递。这一电位也和从其他部位来的电位相加，如果总和超过阈值，则导致神经细胞B的轴索起始部产生活动电势，它从神经细胞B的轴索前端向末端传输，影响下一个神经细胞C。总之，信息的传递，不论在神经细胞间或在神经细胞内，都是通过电位的变化来进行的。

图2 神经网络

我们把不接触任何末稍的那些神经细胞称为输入神经细胞，其余的神经细胞则称为内在神经细胞。

在相等的时间间隔内（我们把它取为整数，作为时间尺度，并且对于一个给定的神经网络的所有神经细胞均取同样的时间间隔），一个神经网络里的每一个神经细胞，或者是处于受到兴奋的状态，或者是处于不受兴奋的状态（即安静的状态）。对于输入神经细胞来讲，在任一瞬间t，受到兴奋或不受兴奋，完全取决于这个神经网络的外部条件。我们可以假定，感受器官系与某一些输入神经细胞相接触。并且在周围环境具备适当条件的情况下，可在瞬间t使某一输入神经细胞受到刺激。倘若我们赋与兴奋性的神经末稍一个正数、赋与抑止性的神经细胞一个负数作为它们的权；那么，对于内在神经细胞来讲，在任一瞬间t，它受到兴奋或不受兴奋取决于在瞬间$t-1$它的所有神经末稍的权的总和是否超过这个内在神经细胞的阈值。

这是一个高度简化了的神经生理学的描述，它导致麦克卡洛克——比脱斯关于神经细胞的一个模型。

定义1·1 一个模块（或形式化的神经细胞）具有m个输入x_1，…，x_m（$m\geqslant 1$）和一个输出d。除了这$m+1$个数之外，还有一个阈值θ，以及权w_1，…，w_m，其中权w_i与x_i相对应。这一模块作用于一离散时间尺度$t=1$，2，3，4…之上。在$t=n+1$时它的输出兴奋与否按以下的规则决定：

倘若引入符号

$m(t)=1$表示“m在时间t兴奋”，$m(t)=0$表示“m在时间t不兴奋”，（在此m可以是一个神经细胞的输入或输出），那么

$$d(n+1)=1 \quad \text{当且仅当} \quad \Sigma w_i x_i(n)\geqslant\theta。$$

有了这样一个十分简单的神经细胞的模型之后，我们可以立刻定义一个神经网络的模型。

定义1·2 一个模块网络是一组模块的集合，它们相互联结，有共同的时间尺度。一个模块的一个输出可以分裂为若干条线，这些线的一部分（或全体）可以与其他模块的输入相联

结。一个输出可以引导至若干个模块，但是一个模块的每一个输入只能来自一个（最多一个）模块的输出。

如图3的例，它有3根输入线l_0，l_1，l_2，它们不与任何模块的输出相联；它有4根输出线p_0，p_1，p_2，p_3，它们不与任何模块的输入相联。

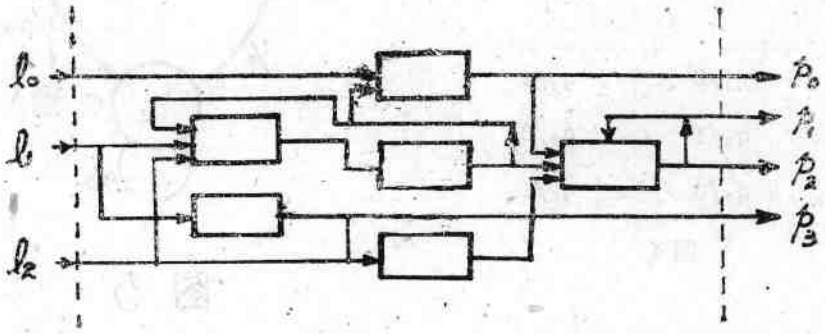


图3. 一个简单的模块网络

二、另一个数学模型——有限自动机

工程师认为作出了一个系统的模型，倘若他能确实构造一些器械来模拟这一系统的行为。另一方面，数学家认为他作出了一个系统的模型，倘若他能用一些准确的数学定义与公理“捕获”了这一系统的某些性质，对这一“形式”模型（亦即数学模型）能推导出更多的性质，解释这系统的原有的某些性质，并予言新性质。在这一意义上说，模块网络虽是脑的一个数学模型，但它更类似于一个工程模型，因为我们可以用一些电子元件来实现它。在数学上，它是不很方便于“使用”的。为此，我们引入另一个更抽象的数学模型——有限自动机。并且在此说明，任意一个模块网络都是一个有限自动机。

定义 2·1 自动机是一系$A=(Q, \Sigma, O, \delta, \lambda, q_0, F)$其中$Q$是有限的状态字母集；$\Sigma$是有限的输入字母集；$O$是有限的输出字母集；$\delta$是笛卡儿积集$Q\times\Sigma$到$Q$的映射，称为下一状态函数；$\lambda$是$Q\times\Sigma$到$O$的映射，称为输出函数；$p_0\in Q$是$Q$的某一状态，称为开动状态；$F\subseteq Q$是$Q$的一个子集，称为终止集。

将自动机看做一个发生器，它是一个从一串输入导致一串输出的一个机器的数学描述：倘若它在时间t处于状态q（$q\in Q$），并且接受一个输入x（$x\in\Sigma$），那么它在时间$t+1$就处于状态$\delta(q, x)$而且发射输出$\lambda(q, x)$。

函数δ和λ可以用一个表或一个有向图来表示。

（Ⅰ）用表来表示：以第i行记状态q_i（$q_i\in Q$），以第j列记输入x_j（$x_j\in\Sigma$），第i行第j列的表值记$\delta(q_i, x_j)/\lambda(q_i, x_j)$。

（Ⅱ）用有向图来表示：每一个状态用一个节点标记；从节点q_i到节点q_j用一有向边联结并在边上标记x/y，当且仅当$\delta(q_i, x)=q_j$而且$\lambda(q_i, x)=y$，其中$x\in\Sigma$，$y\in O$。

例：$Q=\{q_0, p_1, q_2\}$，$\Sigma=\{0, 1\}$，$O=\{0, 1\}$，它的函数δ和λ如图4的表所表示或者由图5的有向图所表示。

q \ X	δ(q, x)／λ(q, x)	
	0	1
q_0	$q_1/0$	$q_1/1$
q_1	$q_0/1$	$q_2/0$
q_2	$q_2/0$	$q_0/1$

图 4

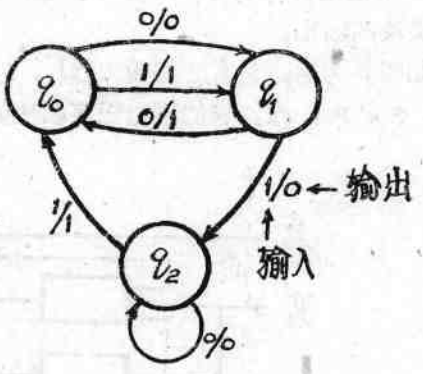


图 5

Σ中的符号所作成的符号串（我们称之为“字”）连同空字Λ构成一集Σ*，映射δ和λ的定义域可以扩充到$Q\times\Sigma^*$上，递归定义如下：对所有的x，$x'\in\Sigma^*$，

$\delta(q, \Lambda)=q$，$\delta(q, xx')=\delta(\delta(qx), x')$；

$\lambda(q, \Lambda)$无定义，$\lambda(q, xx')=\lambda(\delta(q, x), x')$当$x'$不为空字Λ。

由一输入符号串（Σ*中的字）就导致一输出符号串，亦即O中符号作成的一个符号串。我们用以下的图 6 表示自动机的这种功能：

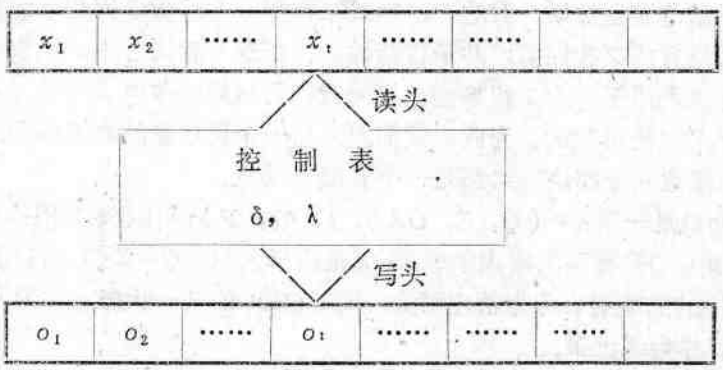


图 6

它有两条有限长的带，带上有若干个方格。一条是输入带，每方格存放一个输入字母；一条是输出带，每方格存放一个输出字母。它有一个读头和一个写头。它自左向右每一节拍走一格，走至最右端，自动机即行停止。它有一个控制器，内部存放一张状态表δ和一张输出表λ。它根据当前的状态和当前注视的输入字母，根据δ决定下一状态，根据λ在输出带上写下输出字母。

自动机还可以作为一个接受器，它可以用来识别Σ*的一个子集中的字。这时我们可以将输出字母集O和输出函数λ忽略不管。这时它只有一条输入带而无输出带，只有一个读头而无写头；它从开动状态q_0开始注视输入带上最左端的一个方格，自左向右每一节拍走一

格直至最右端的一个输入符为止。如果停止时处于状态q而$q\in F$，那末我们说输入带上的字x被接受。若$q\in F$，那么我们说x不被接受。若将Σ^*中被自动机A所接受的字所构成的集合记为$accept(A)$，那么

$$accept(A)=\{x\mid x\in\Sigma^*\text{并且}\delta(q_0,x)\in F\}。$$

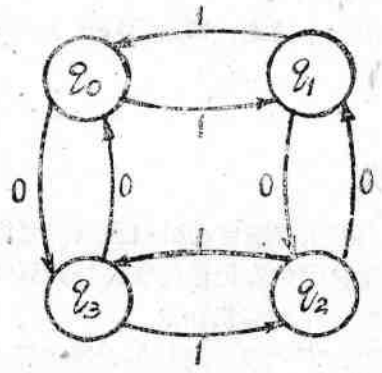


图 7

例 $A=(Q,\Sigma,\delta,q_0,F)$，

$Q=\{q_0,q_1,q_2,q_3,\}$，

$F=\{q_0\}$，$\Sigma=\{0,1\}$，

δ由左边的有向图所表示，有向边上的标记表示输入字母.

$accept(A)=\{x\mid x\in\Sigma^*$，$x$含偶数个0与偶数个1$\}$。

为什么说任意的一个模块网络是一个自动机呢？可以这样看出：假设模块网络N有g个模块，m个输入线，r个输出线。倘若我们知道每一根输入线是接通或断开(兴奋或者不兴奋)，那么我们说知道了这个模块网的输入。可以对这m个输入线以各种可能的方式赋与值0（断开）或值1（接通），因此有2^m个输入；相仿，有2^r个输出。

N的每一模块$t+1$时兴奋与否可以由此模块在t时的各输入来决定。而此模块之输入，有的来自输入线，有的来自其他模块的输出。因此，这一模块在$t+1$时兴奋与否由整个模块网在t时的状态与输入来决定。因而整个模块网在$t+1$时的状态由它在t时的状态与输入来决定。相仿，N的每一根输出线在$t+1$时兴奋与否（亦即整个模块网在$t+1$时的输出）由它在t时的状态与输入来决定。因此，模块网N是一自动机。

值得注意的是，另一方面，任意一有限自动机可以由模块网来模拟。我们按以下的方式构造这个模块网。

设有限自动机$A=(Q,\Sigma,O,\delta,\lambda)$，其中

$Q=\{q_1,\cdots,q_l\}$，$\Sigma=\{x_1,\cdots,x_n\}$，$O=\{o_1\cdots,o_m\}$。

我们作一模块网N，有n个输入线$h_1,\cdots,h_n$（因而有2^n个输入）与m个输出线$p_1,\cdots p_m$（因而有2^m个输出）。用N中的输入$\overline{x_i}$与A的输入x_i相对应；$\overline{x_i}$是N中这样的一个输入：即输入线h_i兴奋而其它的输入线均不兴奋。用N中的输出$\overline{o_i}$与A的输出o_i相对应；$\overline{o_i}$是N中这样的一个输出：即输出线q_i兴奋而其它的输出线均不兴奋。

我们构造$ln+m$个模块，其中ln个模块标记为$\langle k,j\rangle$（$1\leqslant k\leqslant l$，$1\leqslant j\leqslant n$），另外m个模块标记为$\langle i\rangle$（$1\leqslant i\leqslant m$）。模块$\langle k,j\rangle$对应A的状态q_k与输入x_j，模块$\langle i\rangle$对应于A的输出o_i。

我们这样安排模块之间的联结，使得模块$\langle k,j\rangle$在$t+1$时兴奋当且仅当A处于状态k同时接受输入x_j；模块$\langle i\rangle$在$t+1$时兴奋当且仅当A在t时发射输出o_i。

为了达到这一目的，我们只须按照以下的方式安排。设集合$\{k_1, \cdots, k_{n(x)}\} = \{(s, t) | \delta(q_s, x_i) = q_k\}$。令模块《$k, j$》有函数$(k, j)(t+1)$当且仅当$h_j(t) \wedge [k_1(t) \vee \cdots \vee k_{n(k)}(t)]$，（在此$\wedge$表示“同”，$\vee$表示“或”，$k_1(t)$表示$k_1$在$t$时兴奋）。设$\{\overline{k_1}, \cdots, \overline{k_{m(k)}}\} = (s, t) | \lambda(q_s, x_i) = o_k\}$，令模块《$i$》有函数$(i)(k+1)$当且仅当$\overline{k_1}(t) \wedge \cdots \wedge \overline{k_{m(k)}}(t)$。再令模块《$i$》发射出输出线$p_i$。

用N中的状态$\overline{q_s}$与A的状态x_s相对应。$\overline{q_s}$是N中的这样的一个状态：即诸模块《s, j》$(1 \leqslant j \leqslant n)$中有一且仅一个兴奋。

不难看出这时模块网N模拟自动机A。

三、自动机与数字电子计算机

在这一节里，我们说明数字电子计算机如何可以看作是由若干有限自动机相互连结而构成的。因为自动机又可以被模块网络所替代，由此可以看出模块网络有记忆的能力与计算的能力。模块网络虽是人脑的一个十分粗略的模型，但它已经包含了计算机在内。

一般的数字电子计算机通常包含四个部分：输入和输出器、存贮器、逻辑控制器和算术运算器。输入器从输入纸带读入指令或数据，然后将它们转移至存储器中。存储器含有若干个单元，每个单元有一地址，每一单元保存一个“字”，它可以是数据或指令。控制器每次从存储器取一条指令然后执行它。倘若是一输入或输出指令，它就将它转至输入器或输出器；如果是一地址数，就送入存储器；如果是一算术运算，就送至运算器；如果是一判断，它就在施行判断之后决定它的下一条指令。

我们用以下的框图描述数字计算机的作用原理。

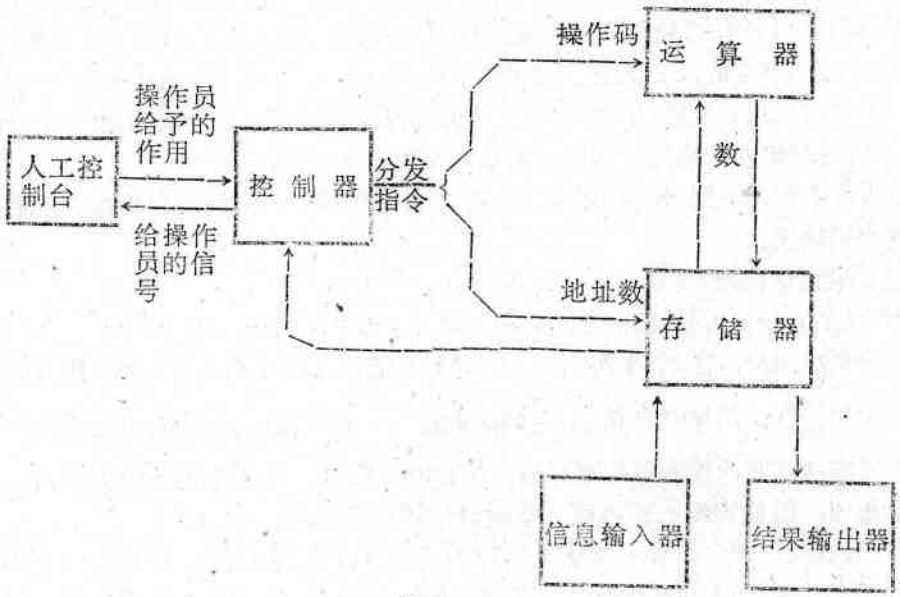


图 8

为了具体，我们假定指令是三地址的，形式如下：

操作码	运算数 1	运算数 2	下一指令

它包含操作码和三个地址，头两个地址告诉控制器在什么地方去找它的运算数，第三地址告诉控制器在什么地方去找它的下一个指令。

现在我们描述如何可将一数字计算机看作是一些自动机连结而成。

A. 存储器（图9）

状态：当存储器的n个单元含n个字x_1，…，x_n时，简记为（x_1，…，x_n）作为它的状态。

输出：有四条线，两根到运算器，一根到输出器，一根到控制器，分别记为$a1$，$a2$，o，l。

输入：有两种形式。

（Ⅰ）（m，b，o），它将字b存入地址m，从状态（x_1，…，x_{m-1}，x_m，x_{m+1}，…）改变到状态（x_1，…，x_{m-1}，b，x_{m+1}，…，），但无输出。

（Ⅱ）（m，o，d），状态不变，但将地址m中的字x_m沿着d输出线（d可为$a1$或$a2$或o或l）输出。

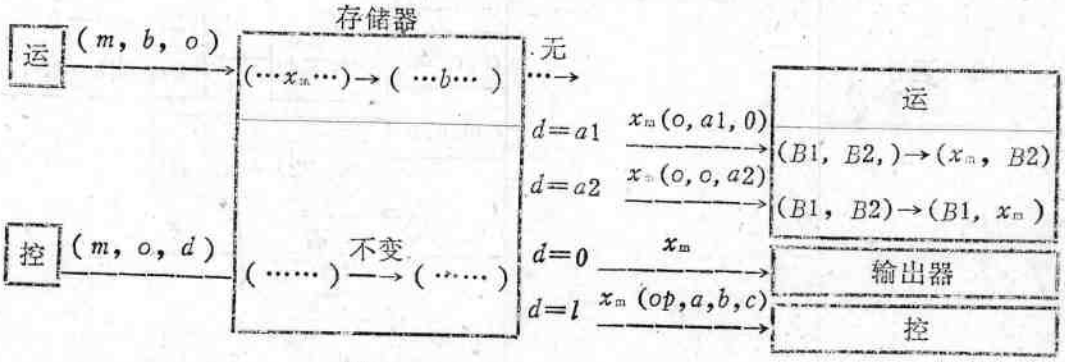


图9

B. 算术运算器（图10）

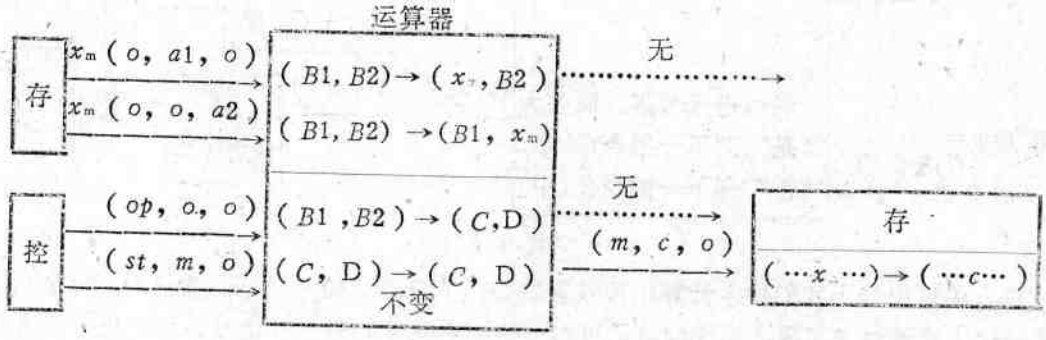


图10

状态：它有两个寄存器，寄存器的内容$B1$，$B2$构成它的状态（$B1$，$B2$）。

输入：有三根输入线，是一三元组（op，$a1$，$a2$）。

输入x_m（o，$a1$，o）来自存储器，将x_m存入第一寄存器，从状态（$B1$，$B2$）变到状态（x_m，$B2$），而无输出。相仿，输入x_m（o，o，$a2$）使状态（$B1$，$B2$）变到（$B1$，x_m）而无输出。

输入（op，o，o）来自控制器，当op为一运算，运算在（$B1$，$B2$）上施行，得到结果C，从而使状态（$B1$，$B2$）变为（C、D）（D可以与$B2$相同），而无输出。当op表示存储st，那么输入（st，m，o）使状态（C、D）不变，而将输出（m，C，o）送存储器。

C。控制器（图11）

它的输入就是来自存储器的指令$opabc$。输入$opabc$使它从原来的状态改变到状态（op，a，b，c）。倘若op是一算术运算，那么它相继发出四个输出，按以下的顺序，一个接着一个：先是（a,o,$a1$）与（b，o，$a2$）至存储器；然后（op，o，o）至运算器，施行运算；最后输出（c、o、l）至存储器，使存储器将下一指令（即地址c中的指令）送控制器。如果op是一存储st，那么它发出两个输出：（st，a,o）至运算器，然后（c，o，l）至存储器。如果op是分枝运算（转移指令）Br，（$Brabc$表示对地址a的内容作一判断，若答案为“是”则下一指令的地址为b，为“否”则下一指令为c），那么在控制器中施行这一判断然后相应地发出（b，o，l）或（c，o，l）至存储器。

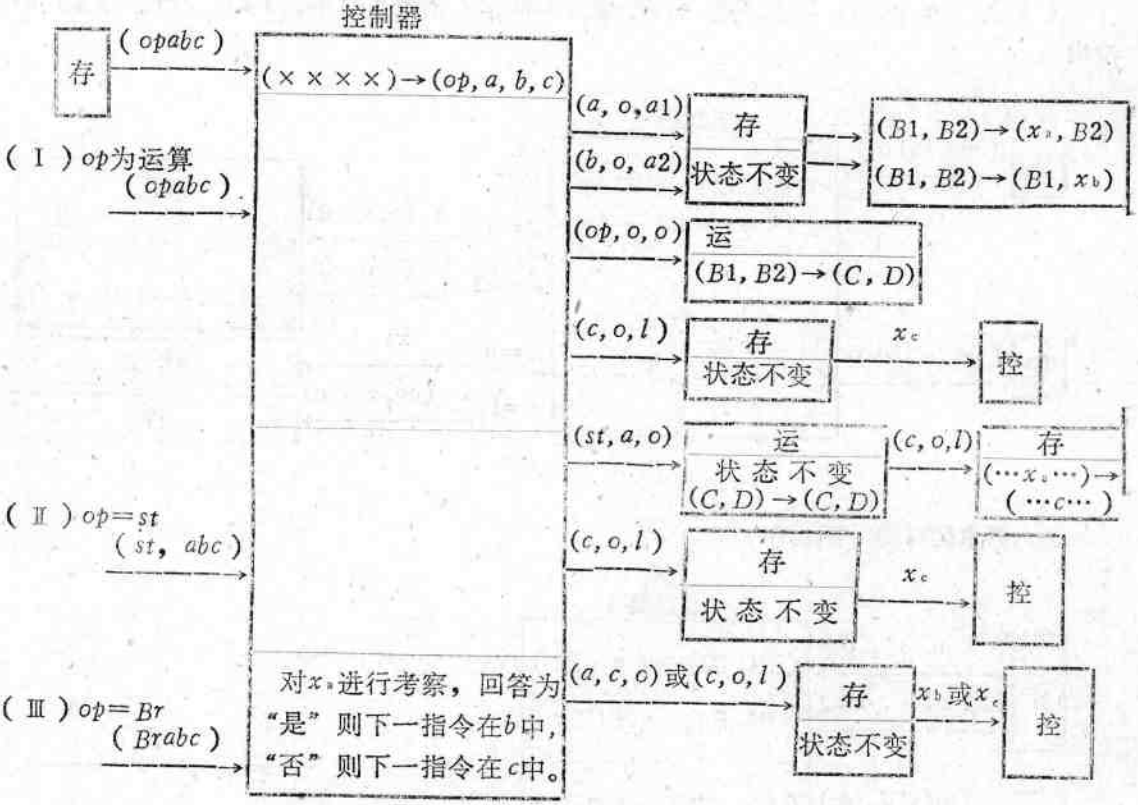


图1 1

以上的讨论告诉我们数字计算机可以看做是几个自动机相互联结而构成的。每个自动机可由一个模块网络来实现，模块网络又可由一些电子元件如磁心存储器、触发器与晶体管构成，虽然这些模块不是严格的麦克卡洛克——比脱斯的神经细胞模型。我们在此所谈到的只是一般概念，而不是机器的具体电路。

前面谈到的自动机，虽有输出，但没有讲到它的输出实际干什么工作。对于不管输出只有识别功能的自动机，它只能识别信息而不能加工信息，并且只能从左向右读去。数字计算机则不然，它不仅能识别信息，而且它的输出可以在它的存储器中写入并更改信息，而且按照

分枝判断改变执行指令的顺序。

四、图 灵 机

图灵（$A\cdot M\cdot$Turing）在1936年引入了图灵机的概念，它被设计作为一个描述过程的数学模型。上面说过，数字电子计算机可以看作是一个自动机，这个自动机有记忆的能力（有存储器）与计算的能力，它不仅可以读而且可以写或更改输入的信息。图灵把这样的自动机作了简化而更具体的规定。

在一个输入字母集Σ（一有穷集）上的图灵机含有三个部分：

1．一条被分成若干单元的输入带。输入带有一个最左单元，但其右端是无穷的。输入带符号集Γ是一有穷集，它包含输入字母集Σ，还包含若干辅助符号的字母集V，还包含一个特殊的空白符号B，所有这些符号是不相同的：$\Gamma=\Sigma\cup V\cup\{B\}$。输入带的每个单元只含一个带符号。

2．一个带头（读写头），每一次注视一个带单元，它可以移动，在每一动作中，带头在它所注视的单元上印刷一个不是空白符B的带符号来代替在那里曾写过的符号，并向左或向右移动一个单元。

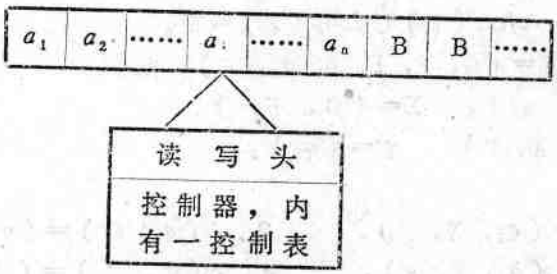


图12

3．控制表，它指明根据机器内部的状态和看到的符号如何决定下一步的动作和机器要进入的状态。假设表示状态的符号集是$Q=\{q_1, q_2\cdots, q_n\}$。控制表由若干指令组成，指令有如下的五重组

$$q_i s_j s_k r q_l$$

的形式，其中r表示符号L或R；它说明机器处于状态q注视符号s则在所注视的单元上打印符号s_k（s_k可以与s_j相同）并按r为L或R而向左或右移动一单元然后进入状态q_l。五重组包括读、写、移位、和改变状态。

对于不同的五重组，它的头两个符号$q_i s_j$不能相同，这就保证机器在任何时候最多只会执行一件事。

现在我们可形式地将图灵机定义如下：

图灵机机$T=(Q, \Sigma, \Gamma, \delta, q_0, F)$，其中

Q是状态的有穷集。

Γ是带符号集，其中有一个是表示空白的符号，记为B。

Σ是输入符号集，它是Γ的真子集，不含符号B。

δ 是下一动作函数，是一映射$Q\times\Gamma\to Q\times(\Gamma-\{B\})\times\{L, R\}$，而对某些变元，$\delta$可不予定义。

q_0是一开动状态，它属于Q。

$F\subseteq Q$是终止状态集。

最初，带上n个最左的单元含有输入集Σ中的符号，剩余的无穷个单元含有空白符号B，换言之，Σ上的一个字（Σ中符号的一个有穷序列）置放在带的左端；这时带头处于开动状态q_0，注视最左的一个单元。当带头按照控制表中的指令执行动作，直至处于一个状态q注视到一个Γ中符号x而$\delta(q, x)$没有定义时，机器便停止。另外，当带头正注视带上最左的一个单元，而应该执行的指令指示它向左移动，这时机器也停止。对于终止集F中任意状态q，不论注视到Γ中的什么符号x，我们均假定$\delta(q, x)$没有定义，因此机器必然停止。

以Σ^*表示Σ上的字连同空字Λ所构成的集合。将一个字$w\in\Sigma^*$作为输入，或者机器在某非终止状态停止，我们说w被接收；或者在某一非终止状态停止，我们说w被拒绝；或者永不停止，我们说w循环。对于以Σ为输入集的图灵机T，所有被接收的字所构成的集合记为$accept(T)$，被拒绝的字所构成的集合记为$reject(T)$，循环的字所构成的集合记为$loop(T)$。显然这三个集合是彼此不相交的，并且

$$accept(T)\cup reject(T)\cup loop(T)=\Sigma^*。$$

例 以下是图灵机$T=(Q, \Sigma, \Gamma, \delta, q_0, F)$，其中

$Q=\{q_0, q_1, \cdots, q_5\}$，　$\Sigma=\{0, 1, \}$，

$\Gamma=\{0, 1, B, X, Y\}$，　$F=\{q_5\}$。

且δ定义如下。

1. $\delta(q_0, 0)=(q_1, X, R)$，　2. $\delta(q_1, 0)=(q_1, 0, R)$，
3. $\delta(q_1, Y)=(q_1, Y, R)$，　4. $\delta(q_1, 1)=(q_2, Y, L)$，
5. $\delta(q_2, Y)=(q_2, Y, L)$，　6. $\delta(q_2, X)=(q_3, X, R)$，
7. $\delta(q_2, 0)=(q_4, 0, L)$，　8. $\delta(q_4, 0)=(q_4, 0, L)$，
9. $\delta(q_4, X)=(q_0, X, R)$，　10. $\delta(q_3, Y)=(q_3, Y, R)$，
11. $\delta(q_3, B)=(q_5, Y, R)$。

除在规则1至规则11之外，δ均无定义。

这个图灵机能接受（识别）Σ^*中所有形$0^n1^n(n\geqslant 1)$的字而拒绝Σ^*中其他的字。亦即

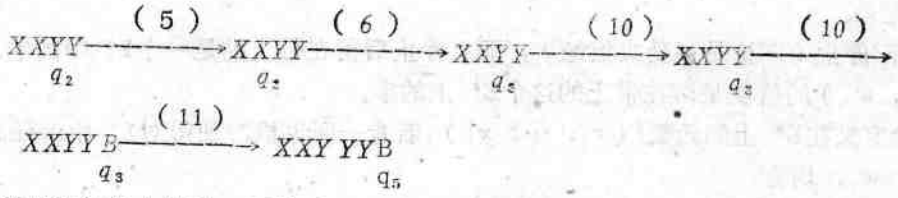
$$accept(T)=\{0^n1^n\mid n\geqslant 1\}，\quad reject(T)=\Sigma^*-accept(T)\text{ 而 }loop(T)=\Phi$$

（空集）。

现在让我们看一看T对输入0011是如何动作的。在以下的序列中，凡被带头注视着的符号下方写上当前的状态，横线上的标记表示使用的δ号数。

$$\underset{q_1}{0011}\xrightarrow{(1)}\underset{q_1}{X011}\xrightarrow{(2)}\underset{q_1}{X011}\xrightarrow{(4)}\underset{q_2}{X0Y1}\xrightarrow{(7)}$$

$$\underset{q_4}{X0Y1}\xrightarrow{(9)}\underset{q_0}{X0Y1}\xrightarrow{(1)}\underset{q_1}{XXY1}\xrightarrow{(3)}\underset{q_1}{XXY1}\xrightarrow{(4)}$$



图灵机T进入终止状态q_5而停止，所以0011是被T所接受的.

指令也可由以下形式箭头所表示：

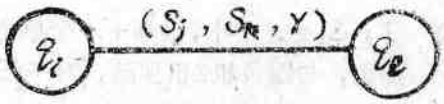


图 13

它表示若机器正处于状态q_i，带头注视符号s_j，则在所注视的单元上打印符号s_k，然后根据r为L或R而向左或向右移动一单元，然后转入状态q_l，应用这一表示方法，控制表就可表为一有向图。例如上例就可表为如下的图。

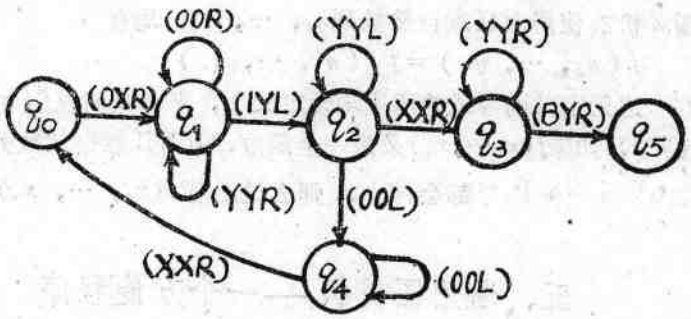


图14

从图灵机的传递图不难看出，图灵机与无输出的有限自动机有三点区别：它有读写头，不仅能读带上的符号而且能写与更改带上的符号，有限自动机则只有读头而无写头；根据指令的安排，它可以向左和向右移动，可以随时注视某一方格（即某一地址），而有限自动机只能从左向右按顺序读去；由于图灵机的读写头可以随时右移并在带上写符号，所以带的右端所能提供的方格个数不受限制，而有限自动机的带上符号只能是在输入时写上去的有限个。

图灵机不仅可以作为一个识别装置，而且可以作为一个计算装置：它定义一个函数。

设Z是输入字母集$\Sigma_1 \cup \{*\}$上的一个图灵机。对任意给定的正整数r，有一个定义在Σ_1^*上的函数$F_r(x_1, \cdots, x_r)$与图灵机Z相伴随。$F_r(w_1, \cdots, w_r)$将Σ_1^*中的字的序列$w_1, \cdots, w_r$映射到Σ_1^*中的字；它按以下的方法定义：以$w_1 * \cdots * w_r$作为输入字输入图灵机Z。

（Ⅰ）若Z不停止，或虽停止而停止后留在带上的不是一个Σ_1^*中的字，则$F_r(w_1,$

…，w_r）无定义。

（Ⅱ）若Z停止（不论是接受或拒绝），而且停止后留在带上的是一个Σ_1^*中的字，那么$F_z(w_1, \cdots, w_r)$的值就是留在带上的这个Σ_1^*上的字。

对于一个定义在Σ_1^*上的函数$f(x_1, \cdots, x_r)$，若有一图灵机Z，使得对Σ_1^*中的任意字的序列$w_1, \cdots, w_r$，均有

$$f(w_1, \cdots, w_r) = F_z(w_1, \cdots, w_r)$$

（在此等式的一边无定义时则等式的两边均无定义），那么我们说函数$f(x_1, \cdots, x_r)$是图灵可计算的。

特别，若Σ_1仅含“单位符”1，$\Sigma_1 = \{1\}$，以$n+1$个单位符1表示自然数n（例如1111表示数3而1表示数0），那么，与图灵机Z相伴随，对于任意给定的正整数r有一个定义在自然数上的函数$F_z(x_1, \cdots, x_r)$。对自然数序列$n_1, \cdots, n_r$，$F_z(n_1, \cdots, n_r)$按以下的方式定义：以$1^{n_1+1} * \cdots * 1^{n_r+1}$输入Z（例如对应于序列2，0，3，输入字就是111＊1＊1111），

（Ⅰ）若Z不停止，则$F_z(n_1, \cdots, n_r)$无定义，

（Ⅱ）若Z停止（不论是接受或拒绝），那么$F_z(n_1, \cdots, n_r)$的值就是留在带上的1的个数（注意：不是留在带上的1的个数减一）。

对于一个定义在自然数上的函数$f(x_1, \cdots, x_r)$，如果有一在输入字母集Σ（Σ含符号1与＊）上的图灵机Z，使得对任意自然数列$n_1, \cdots, n_r$，均有

$$f(n_1, \cdots, n_r) = F_z(n_1, \cdots, n_r)$$

（在此当等式的一边无定义则等式的两边均无定义），那么我们说$f(x_1, \cdots, x_r)$是图灵可计算的或部分递归的。如果$f(x_1 \cdots, x_r)$又是一全函数，即f对任意自然数列$n_1 \cdots, n_r$均有定义（亦即Z对任意输入$1^{n_1+1} * \cdots * 1^{n_r+1}$都会停止），那么说函数$f(x_1, \cdots, x_r)$是递归的。

五、通用图灵机——一个万能程序

有效过程是一个直观的概念，它意味着能机械地实现的一串指令之一有穷序列。例如，用欧几里德（Euclid）算法寻找两个整数的最大公约数就是一个有效过程。任何一个能够由数字电子计算机完成的计算是由一串纯粹机械的规则所控制的，因此是一有效过程。在有些情况，我们虽然还不知道如何编写一个程序让机器去实行我们要做的计算，但是我们有强烈的直感，认为对这一计算的有效过程是一定存在的；并且，经过细致的考虑与安排也是一定可以编写出这个程序的。然而，随意抛掷一个铜元，我们就不认为有一个有效过程能予言抛掷的结果是正面还是反面。

根据图灵机的定义，显然任何能用图灵机描述的计算都是可以机械地实现的。另一方面，也能证明，任何在现代数字计算机上能实现的计算都能用图灵机描述。有效过程是一个直观的概念，这在数学上来说是不精确的；因而图灵机就被建议作为与有效过程等同的一个精确的数学概念，这就是邱吉（*Church*）假设：

“任何一个有效计算，亦即一符号串到符号串的映射，总能由一图灵机来执行。”

对这一假设当然不能给出一个形式的证明，正因为有效过程这一概念本身就是一个直观

的概念！我们只能说，迄今为止，任何一个在直观上为显然的有效过程，总能设计一个图灵机来实现这一过程。凡是一过程在直观上认为是有效的、能行的，那么我们就承认有一适当的图灵机存在，能完成这一过程。在以下要讨论的几种情况，都可实际作出具体的图灵机，不过在这里就不详细写出了。

以下我们指出，存在一个通用图灵机U，它能模拟任意一个图灵机，包含它自身在内。

设T_1是输入符号集$\Sigma_1=\{a_1, \cdots, a_n\}$上的图灵机，我们可以只用符号0与1将$\Sigma_1^*$中的字编码；例如以$1^i$（$i$个1）与$a_i$对应，以$1^i01^j$与$a_ia_j$对应，等等。于是有一图灵机$T_2$，它以$\Sigma=\{0, 1\}$作为输入符号集，它的行为与$T_1$完全等同，只须将$\Sigma_1^*$中的字用$\Sigma$上的符号编码。因此我们可以假定所有图灵机的输入符号集就是$\Sigma=\{0, 1\}$，带符号集$\Gamma=\{0, 1, B\}$，B表示空白。

Σ^*中的字可以按词典顺序排列，例如头几个字就是Λ，0，1，00，01，10，11，000，001，010，011，…，第一个字是空字Λ，第二个字是0，…，等等。一般规则是：先按字长比较顺序；有相同字长的字，将符号看作是二进制数，再按数的大小比较顺序。例如字011，它有3位，按二进制表示，11为3，因此字011是第$2^3+3=11$个。又如000是第$2^3=8$个。反之，给定任意一个正整数n，我们可以写出第n个字：设$n=2^r+m$（r是n中所含2的最高次幂），那么第n个字有r位；再将m表为二进制字数，于是就可写出这个字。例如$20=2^4+4$，因此第20个字是0100。

带符号集$\Gamma=\{0, 1, B\}$上的图灵机也可按一定的顺序排列。首先，每一个Γ上的图灵机可以按以下的规定编码为$\{0, 1, C, L, R\}$上的字。

先将状态分为n个横行与3个纵列，每行表示一个状态，而3个列分别对应B，0，1。若$\delta(q_i, a)=(q_j, b, D)$（$D$表$L$或$R$，$a$与$b$为$\Sigma$中的符号0，1），那么表中第$i$行与含符号$a$的列中的内容就编码为$1^jDb$，若$\delta(q_i, a)$无定义则编码为0。先按行编码，每行又按列$B$，0，1的顺序编码。列与列之间用一个$c$隔开，行与行之间用两个$c$隔开，开始与末尾写三个$c$。例如以下的状态表就编码为右边的字母行。

状态	B	0	1
1	—	—	2, 0, R
2	3, 1, L	3, 1, L	2, 1, R
3	4, 0, R	4, 0, R	3, 1, R
4	—	—	—

图15

$cccococ11Ro\ cc$
$111L1c111L1c11R1cc$
$1111Roc1111Roc111R1cc$
$ocococcc$

此外，我们规定q_1为开动状态。对于终止状态q，我们规定$\delta(q, a)$均无定义，无论a为B或0或1；反之若有某一状态q，$\delta(q, B)$与$\delta(q, o)$与$\delta(q, 1)$均无定义，那么q为一终止状态。因此对于非终止状态q，$\delta(q, B)$，$\delta(q, 0)$，$\delta(0, 1)$中至少有一个是有定义的。

分别以0，1，L，R，C对应于数0，1，2，3，4。以5为根基，那么每一个$\{0, 1, L, R, C\}^*$中的字（空字除外），因而每一个图灵机，都可以表为一个自然数。另一方面，对于任意一个自然数n，表为五进制数，就得到一个$\{0, 1, C, L, R\}$中的一

个符号串。它也定义一个图灵机。倘若它不是良构的（即编码不合图灵机的规定），就定义为一个无动作的图灵机。因此，我们可以将图灵机一个一个的枚举出来，即图灵机

$$Z_1,\ Z_2,\ Z_3,\ \cdots\cdots$$

（它们中间可能有重复，即有相同的，但这无关紧要），其中Z_n就是按以上有效过程所决定的第n个图灵机。

给定一对正整数序偶(m,n)，我们可以有效地计算{0，1}上的第m个字X_m，有效地计算第n个图灵机Z_n。因此我们有一个在带符号集{0，1，C，L，R，B}上的通用图灵机U，它可以模拟Z_n的行为。倘若Z_n对输入x_m停止，那么U也停止，并且纸带上在终止后的符号串就是Z_n在终止时纸带上的符号串；倘若Z_n停止在一接受状态，那么U也停止在接受状态；倘若Z_n停止在一拒绝状态，那么U也停止在一拒绝状态；倘若Z_n不停止，那么U也不停止。

特别，设$f(x_1,\cdots\cdots,x_s)$是一个定义在自然数上的可计算的函数，由第n个图灵机所计算，亦即

$$f(x_1\cdots\cdots,x_s)=Fz_n(x_1,\cdots\cdots x_s)$$

那么它也可由通用图灵机U所计算，亦即

$$Fz_n(x_1,\cdots\cdots,x_s)=FU(n,x_1,\cdots\cdots x_s)。$$

六、停机问题——一个不能判定的问题

我们说一个集合是递归的倘若存在一个有效过程能判定任意一元是否属于这个集合。根据邱吉假设，我们可以精确地定义如下：

定义　设Σ为一字母集，Σ^*为Σ中字的集合，$S\subseteq\Sigma^*$为Σ^*之一子集。我们说S是一递归集，倘若存在一输入字母集Σ上的图灵机，它对任意输入字x都保证停止；并且当$x\in S$则被T所接受，当$x\in S$则被T所拒绝。换言之，$\text{accept}(T)=S$，$\text{reject}(T)=\Sigma^*-S$，$\text{loop}(T)=\phi$（空集）。

定义　设集合$S\subseteq\Sigma^*$，我们说S是递归可枚举的，倘若有一Σ上的图灵机（它不一定保证停止），当$x\in S$则被T所接受，当$x\in S$则被T所拒绝或者永不停止。换言之$\text{accept}(T)=S$，$\Sigma^*-S=\text{reject}(T)\cup\text{loop}(T)$。

易见递归集是递归可枚举集的子集。

设$S\subseteq\Sigma^*$为一递归可枚举集，那么有一图灵机T（不一定保证停止）能接受它。对于这样的一个集合S，我们可以将它的元素一个挨一个地产生出来。但在这样做时应当小心。由于Σ^*中的字可以排列为x_1，x_2，…，我们可以逐次将x_1，x_2…，送入图灵机T进行考察，看它们是否被T所接受，因而判定它们是否属于S。然而当某一个x_i输入T时，若T不停止，我们就无法越过x_i去试验第$i+1$个字x_{i+1}。为了克服这一困难，我们设计以下的方法对x_1，x_2，…进行考察。

对任意一对正整数偶(x,y)，以一个正整数

$$z=\tfrac{1}{2}(x+y-1)(x+y-2)+y$$

与之对应；反之，对任意一整数z，我们可以有效地计算(x,y)。z与(x,y)的对应是1—1的。

在T进行计算时，我们将计算分为若干步（比如说，一个五重组作为一步）。对于给定的一个正整数n，设与n对应的正整数偶为(i, j)。在作第n次试验时，将x_i输入图灵机，并且只计算j步。若在j步内机器停止并接受x_i，那么将x_i放入T所产生的字的表中。若在此j步计算完毕之后，机器未接受x_i，那就继续去做第$n+1$次试验。倘若$x_i \in S$，那么一定存在某一个有限的正整数j使得x_i在计算j步后停止并被接受，因而x_i一定会被产生出来。

若一集合$S \in \Sigma^*$能被一图灵机（不一定保证停止）所接受，因而是一递归可枚举集；但是应当注意，S也可能被另一个保证停止的图灵机所接受，因而也是一递归集。有没有一个集合S，能被不保证停止的图灵机所接受，但不能被任何一个保证停止的图灵机所接受呢？换句话说，有没有一个递归可枚举集而不是递归集呢？这回答是肯定的，以下我们要举出一个例子。

引理1 若S为一递归集，那么它的补集也是递归的。

证 设$S \subseteq \Sigma^*$为一递归集，因而有一保证停止的图灵机T接受S。我们可以假定T在接受之后不再继续动作。现在从T构造另一图灵机T_1，我们添加一个状态q，它是T_1的唯一的一个终止状态（接受状态），T_1的规则包含T的所有规则；此外，对T的每一非接受状态q_i，当注视到某一纸带符号x，若$\delta(q_i, x)$无定义因而T停止时，T_1则转移至添加的状态q然后停止。

于是，当T停止于接受状态时，则T也停止；但T所停止的状态不是q因而不接受x。当T停止于非接受状态，因而拒绝x；但T_1则多一动作，转至状态q而停止，因而接受x。因此T_1（它保证停止）所接受的集合恰好就是$\Sigma^* - S$。

引理2 设x_1，$x_2 \cdots$，为有穷输入字母集$\Sigma = \{0, 1\}$上所有字的有效枚举，T_1，T_2，…，是有穷带符号集$\Gamma = \{0, 1, B\}$上所有图灵机的有效枚举。令集合

$$L = \{x_i \mid x_i \text{为} T_i \text{所接受}\}$$

那么L是递归可枚举的，然而它的补集$\overline{L} = \Sigma^* - L$不是递归可枚举的。

证 L可以被一个图灵机T所接受，T的操作如下。应当注意，对于不属于L的字，T不一定停止。

1、给定一个Σ上的字x，我们可以有效过程决定它是第i个字，因而图灵机T可以计算出i，知道$x = x_i$。

2、T产生出来T_i，然后将控制转移至通用图灵机U，将T_i与x_i输入，使U模拟T_i对输入x_i。

3、若T_i对输入x_i停止而且接受，那么T停止而且接受x_i；若T_i停止而拒绝x_i，那么T停止而拒绝x_i；若T_i不停止，那么T不停止。

因此，L是递归可枚举的，因为L被T所接受。

但是补集$\overline{L} = \Sigma^* - L$不能是递归可枚举的。倘若$\overline{L}$是递归可枚举的，那么$\overline{L}$为某一图灵机（比如说$T_j$）所接受：

$$\overline{L} = \{x \mid x \text{为} T_j \text{所接受}\}。$$

因而

$$x \in \overline{L} \Leftrightarrow x \text{为} T_j \text{所接受}。$$

特别，

（1） $$x_j \in \overline{L} \Leftrightarrow x_j \text{为} T_j \text{所接受}。$$

另一方面，由于$\overline{L}$是L的补集，根据定义

$$\overline{L} = \{x_i \mid x_i \text{不为} T_i \text{所接受}\}。$$