中国科学院大学

University of Chinese Academy of Sciences

计算机科学导论

实验报告

姓名：　唐嘉良

班级：　八班

组号：　第六组

学号：　2020K8009907032

实验名称：　图灵机实验

2021 年 4 月 30 日

1 图灵机设计方案

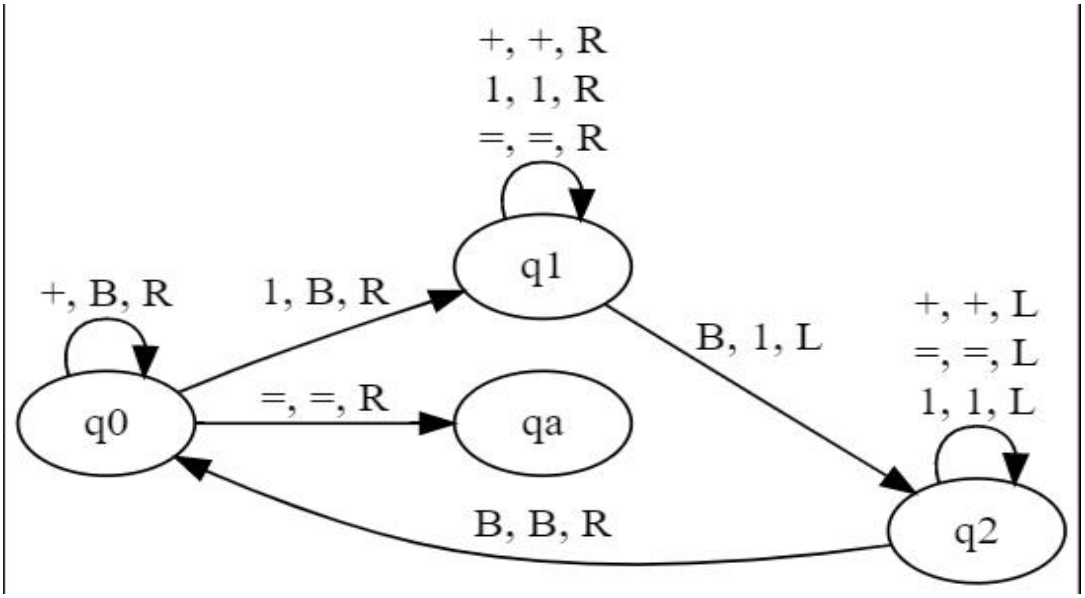
1.1 实验一：设计一个能够完成一进制加法的图灵机

格式要求： 输入：(若干 1)+(若干 1)= ，停机时纸带：...###(若干 1)B###... 其中输入的“若干 1”表示要做加法的一进制数，可能是零个 1；停机时纸带上的“若干 1”作为答案，起始位置应为输入中“=”位置右侧一格，并以 B 作为结尾；# 为Γ中任意字母。

1.1.1 主要设计思路

考虑一进制的进位特点，设计思路是平凡的。我们仅需用读写头将左侧的 1 全部遍历，并且每读到一个 1 就立即在等号左侧写上一个 1，当输入中的 1 全部搬运完后即可得到正确的答案。

1.1.2 状态转移图



1.1.3 状态转移表

q0, 1, B, R, q1
q0, +, B, R, q0
q0, =, =, R, qa
q1, +, +, R, q1
q1, 1, 1, R, q1
q1, =, =, R, q1
q1, B, 1, L, q2
q2, +, +, L, q2
q2, =, =, L, q2
q2, 1, 1, L, q2
q2, B, B, R, q0

1.1.4 模型细论

比较值得注意的是，在该实验中，我们首先将“+”之前的所有 1 搬运到右侧，再把“+”号后面的所有 1 搬运到右侧，期间把所有已经搬运过的 1

全部改成 B，并在左侧的 1 已经完成搬运之后将“+”号也改成 B，从而方便在进入下一个搬运循环的时候能够一直顺利过渡（因为初始状态时输入的左侧就是 B），不需要再添加新的状态。否则相当于在无必要的情况下强行将已经搬运的 1 特殊化，这会增加代码的长度和复杂程度，根据奥卡姆剃刀原理，我们选择了上述改写方法。

整体代码分为两大部分：读取阶段、填写阶段。

1. 读取状态说，如果读到 1，那么会将其改为 B 并进入填写状态，跳到“=”后面写上 1，再进入返回状态，向左一路读到 B 并再次进入读取状态；如果读取状态独到了“+”，那么预示着我们已经遍历了加号前面的所有 1，为了读取加号后面的 1，我们依然把“+”改成 B 并右移一格，但仍保持读取状态不变，这样就能接着遍历“+”后面的所有 1。

2. 计算阶段是紧跟每一次读取的，在读取状态下在等号后面第一个非 1 的数位填上 1 即可。最后容易知道，读写头会在读取状态读到“=”，此时我们已经完成了计算，因而直接跳到 qa 并停机即可。

在设计该图灵机时，没有遇到任何困难。

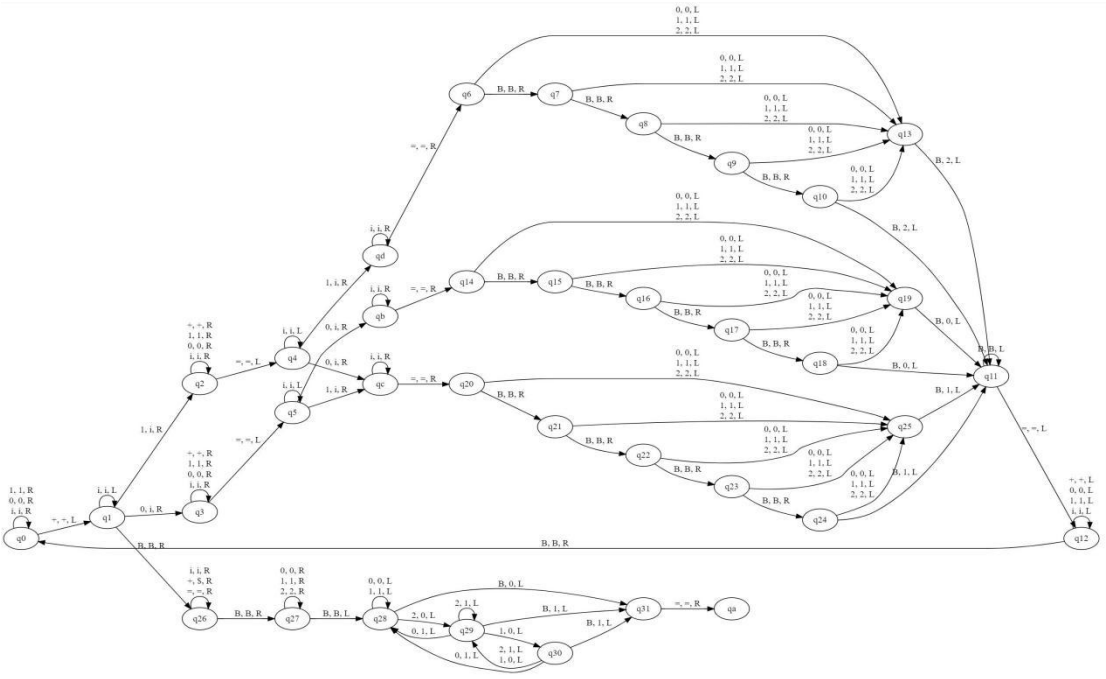
1.2 实验二：设计一个能完成四位数二进制加法的图灵机

格式要求：输入：(四位 0 或者 1)+(四位 0 或者 1)= ，停机时纸带：…###(五位 0 或者 1)B###... 输入的“四位 0 或者 1”表示要做加法的二进制数；停机时纸带上的“五位 0 或者 1” （结果只有 4 位时注意加前导 0）作为答案，起始位置应为输入中“=”位置右侧一格，并以 B 作为结尾；# 为 Γ 中任意字母。

1.2.1 主要设计思路

我们首先完成直接的加法而不考虑进位（例如对应位上 1+1 直接用 2 表示），并在等号右边对应位上填写计算好的数字，此时该数由 0、1、2 组成，并且我们会在等号后面孤立一个空数位（以便下面进行进位），最后我们在等号右侧再用一个循环来考虑进位，并得到最终的五位数答案。

1.2.2 状态转移图



1.2.3 状态转移表

q0,1,1,R,q0
q0,0,0,R,q0
q0,i,i,R,q0
q0,+,+,L,q1
q1,i,i,L,q1
q1,0,i,R,q3
q1,1,i,R,q2
q1,B,B,R,q26
q2,+,+,R,q2
q2,1,1,R,q2
q2,0,0,R,q2
q2,i,i,R,q2
q2,=,=,L,q4
q3,+,+,R,q3
q3,1,1,R,q3
q3,0,0,R,q3
q3,i,i,R,q3
q3,=,=,L,q5
q4,i,i,L,q4
q4,0,i,R,qc
q4,1,i,R,qd
q5,i,i,L,q5
q5,0,i,R,qb
q5,1,i,R,qc
qb,i,i,R,qb
qb,=,=,R,q14
qc,i,i,R,qc
qc,=,=,R,q20
qd,i,i,R,qd
qd,=,=,R,q6
q6,B,B,R,q7
q6,0,0,L,q13
q6,1,1,L,q13
q6,2,2,L,q13
q7,B,B,R,q8
q7,0,0,L,q13
q7,1,1,L,q13
q7,2,2,L,q13
q8,B,B,R,q9
q8,0,0,L,q13
q8,1,1,L,q13

q8,2,2,L,q13
q9,B,B,R,q10
q9,0,0,L,q13
q9,1,1,L,q13
q9,2,2,L,q13
q10,B,2,L,q11
q10,0,0,L,q13
q10,1,1,L,q13
q10,2,2,L,q13
q11,B,B,L,q11
q11,=,=,L,q12
q12,+,+,L,q12
q12,0,0,L,q12
q12,1,1,L,q12
q12,i,i,L,q12
q12,B,B,R,q0
q13,B,2,L,q11
q14,B,B,R,q15
q14,0,0,L,q19
q14,1,1,L,q19
q14,2,2,L,q19
q15,B,B,R,q16
q15,0,0,L,q19
q15,1,1,L,q19
q15,2,2,L,q19
q16,B,B,R,q17
q16,0,0,L,q19
q16,1,1,L,q19
q16,2,2,L,q19
q17,B,B,R,q18
q17,0,0,L,q19
q17,1,1,L,q19
q17,2,2,L,q19
q18,B,0,L,q11
q18,0,0,L,q19
q18,1,1,L,q19
q18,2,2,L,q19
q19,B,0,L,q11
q20,B,B,R,q21
q20,0,0,L,q25

q20,1,1,L,q25
q20,2,2,L,q25
q21,B,B,R,q22
q21,0,0,L,q25
q21,1,1,L,q25
q21,2,2,L,q25
q22,B,B,R,q23
q22,0,0,L,q25
q22,1,1,L,q25
q22,2,2,L,q25
q23,B,B,R,q24
q23,0,0,L,q25
q23,1,1,L,q25
q23,2,2,L,q25
q24,B,1,L,q11
q24,0,0,L,q25
q24,1,1,L,q25
q24,2,2,L,q25
q25,B,1,L,q11
q26,i,i,R,q26
q26,+,$,R,q26
q26,=,=,R,q26
q26,B,B,R,q27
q27,0,0,R,q27
q27,1,1,R,q27
q27,2,2,R,q27
q27,B,B,L,q28
q28,0,0,L,q28
q28,1,1,L,q28
q28,2,0,L,q29
q28,B,0,L,q31
q29,0,1,L,q28
q29,2,1,L,q29
q29,1,0,L,q30
q29,B,1,L,q31
q30,0,1,L,q28
q30,2,1,L,q29
q30,1,0,L,q29
q30,B,1,L,q31
q31,=,=,R,qa

1.2.4 模型细论

思考该题时，一开始最令我头疼的就是进位问题，因为相比这题的数据读取部分和转填部分，带有进位的运算部分使得我们不得不在每次计算时考虑进位的问题，这样是否进位对应不同状态，而不同计算数值又对应下一位计算是否进位，是一个较第一题复杂很多的递进约束结构。

对此，我采用了一点技巧，也就是每次计算都不考虑进位，而在每一位数值都运算完之后统一用一个循环考虑进位，这样做最成功之处在于将进位与加法两大操作彻底分离，使得整体结构变得清晰，难以出错。

全部代码主要分为两大阶段：计算阶段和进位阶段（计算阶段期间的回归阶段是朴素的，在此不去赘述）

1. 计算阶段下，首先往前读到“+”或者“i”就返回一格读取第一个数尚未计算过的部分的最后一位并将其改为 i，而后直接用同样方式读取第二个数尚未计算过的部分的最后一位，我们断言一个事实：这样读取的对应位相加只可能是“1+1”、“1+0”、“0+1”或“0+0”，他们分别对应 2、1、1、0 这三种结果，此时进入填写状态。按照结果，其分为三种填写状态（体现在状态转移图中，就是那明显的三个“羽翼”），每种状态下都先向右读到等号，再通过几个连续的状态找到等号之后对应数位并填写对应数值（规定第一个数值填在等号后面第五位，以“开辟”出所求的五位数结果的位置）。

2. 计算阶段结束之后，我们已经在等号右边得到了一个四位数，且该数与等号之间还有一个“架空数位”为 B。于是，我们正式进入进位阶段。首先读四位数最后一位，如果是 2 就进入“进位状态”，如果是 1 或 0 则继续读倒数第二位，如果是 2 则进入“进位状态”.....如果读到第一位都没有出现 2，那么直接在架空数位上填写 0 得到最终答案；如果有进位状态出现，那么在进位状态下，读前面一位数字并且：将 0 改为 1 并退出进位状态然后读下一位；1 改为 0 并保持进位状态然后读下一位；2 改为 1 并保持进位状态然后读下一位。如果在一般状态读到 B 则改为 0 并停机，如果在进位状态读到 B 则改为 1 并停机。

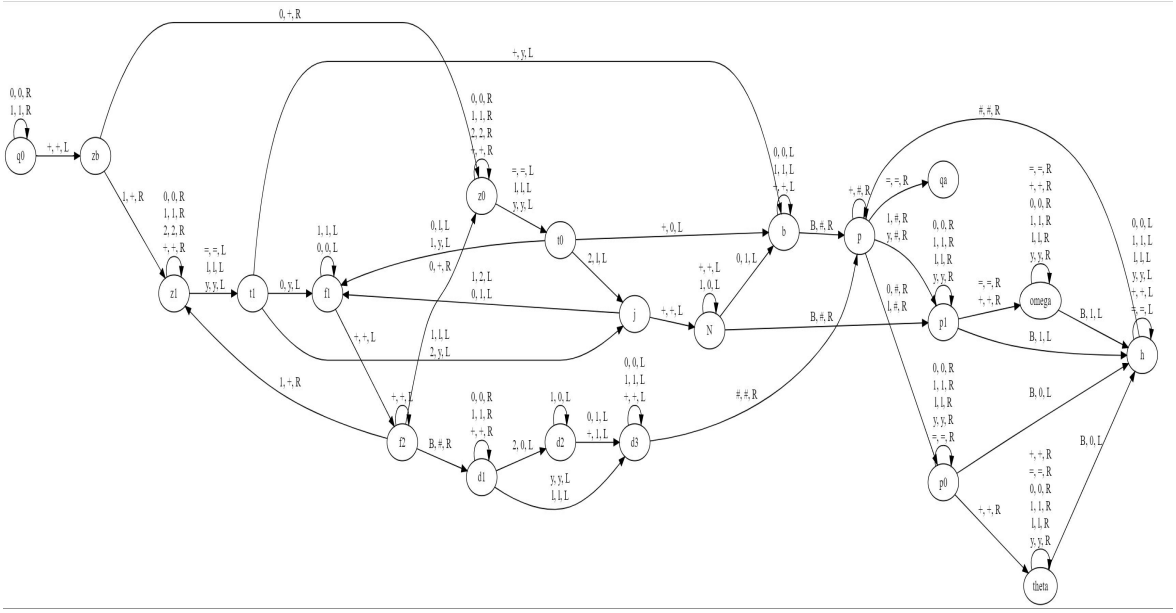
1.3 实验三：设计一个能完成任意位二进制加法（不允许前导零）的图灵机

格式要求：输入：(若干 0 或者 1)+(若干 0 或者 1)=，停机时纸带：…###(若干 0 或者 1)B###...，输入的“若干 0 或者 1”表示保证不含前导 0；每一个操作数的长度大于等于 1。停机时纸带上的“若干 0 或者 1”作为答案，不应含有前导 0，起始位置应为输入中“=”位 置右侧一格，并以 B 作为结尾；# 为 Γ 中任意字母。

1.3.1 主要思路

第三问的输入数值特殊性决定了我们不能够再次采用第二问的思路。于是我另辟蹊径，考虑到答案位数不能被提前确定，我们先在等号左侧进行运算（包括进位），然后再将最终的答案转填到等号右侧即可。

1.3.2 状态转移图



1.3.3 状态转移表

q0,0,0,R,q0	j,1,2,L,f1	p0,y,y,R,p0	h,l,l,L,h
q0,1,1,R,q0	j,0,1,L,f1	p0,+,+,R,theta	h,y,y,L,h
q0,+,+,L,zb	j,+,+,L,N	p0,=,=,R,p0	h,+,+,L,h
zb,1,+,R,z1	f1,1,1,L,f1	p0,B,0,L,h	h,=,=,L,h
zb,0,+,R,z0	f1,0,0,L,f1	theta,B,0,L,h	h,#,#,R,p
z0,0,0,R,z0	f1,+,+,L,f2	theta,+,+,R,theta	d1,0,0,R,d1
z0,1,1,R,z0	f2,+,+,L,f2	theta,=,=,R,theta	d1,1,1,R,d1
z0,2,2,R,z0	f2,1,+,R,z1	theta,0,0,R,theta	d1,+,+,R,d1
z0,+,+,R,z0	f2,0,+,R,z0	theta,1,1,R,theta	d1,2,0,L,d2
z0,=,=,L,t0	f2,B,#,R,d1	theta,l,l,R,theta	d1,y,y,L,d3
z0,l,l,L,t0	N,0,1,L,b	theta,y,y,R,theta	d1,l,l,L,d3
z0,y,y,L,t0	N,+,+,L,N	p1,0,0,R,p1	d2,1,0,L,d2
z1,0,0,R,z1	N,1,0,L,N	p1,1,1,R,p1	d2,0,1,L,d3
z1,1,1,R,z1	N,B,#,R,p1	p1,l,l,R,p1	d2,+,1,L,d3
z1,2,2,R,z1	b,0,0,L,b	p1,y,y,R,p1	d3,0,0,L,d3
z1,+,+,R,z1	b,1,1,L,b	p1,=,=,R,omega	d3,1,1,L,d3
z1,=,=,L,t1	b,+,+,L,b	p1,+,+,R,omega	d3,+,+,L,d3
z1,l,l,L,t1	b,B,#,R,p	p1,B,1,L,h	d3,#,#,R,p
z1,y,y,L,t1	p,+,#,R,p	omega,B,1,L,h	
t0,0,l,L,f1	p,0,#,R,p0	omega,=,=,R,omega	
t0,1,y,L,f1	p,l,#,R,p0	omega,+,+,R,omega	
t0,2,l,L,j	p,1,#,R,p1	omega,0,0,R,omega	
t0,+,0,L,b	p,y,#,R,p1	omega,1,1,R,omega	
t1,0,y,L,f1	p,=,=,R,qa	omega,l,l,R,omega	
t1,1,l,L,j	p0,0,0,R,p0	omega,y,y,R,omega	
t1,2,y,L,j	p0,1,1,R,p0	h,0,0,L,h	
t1,+,y,L,b	p0,l,l,R,p0	h,1,1,L,h	

1.3.4 模型细论

拿到实验三的题目首先的思路是和实验二一样的，就是按照人脑计算的方法，一位一位计算进位输出。但是由于是任意位数的二进制数，容易发现输出的答案位数不能在开始时被确定，那么必须换一种思路。像第二题一样，加出一个数字就去等号之后打印一个数字这样的方法是无法操作的。自然地，我们想到，先在等号左边进行计算，加完后再转移到右边。于是，第三题的读取、进位和打印操作顺序和实验二并不一样，实验三是读取和进位放入一个循环中，而打印放入另一个循环，在第一个循环完成遍历后进入第二个循环。

代码总共分为三个阶段：读取进位阶段，转接阶段和打印阶段。

1. 读取进位阶段。这里的操作是直接将前一个二进制数字加到后一个二进制数字上。这样的读取阶段和实验二完全相同。从右对齐数字对应位相加，但是有做符号区别。并且在进到前一位时候容许有 2 的暂时出现（这里的思想和第二问有异曲同工之妙）。断言读取情况如下：0-0，0-2，1-1 情况写下 L（汉字“零”的拼音首字母），0-1，1-0，1-2 情况写下 Y（汉字“一”的拼音首字母），同时 0-0，0-1，1-0 情况直接进入准备返回的 f1 状态，而 0-2，1-1，1-2 则进入进位状态，然后读写头向左一位将那里的 0 改写为 1，1 改写为 2，再进入返回操作。完成这个小循环回去进行下一轮的对应位置读取相加进位。

2. 转接阶段。转接部分是比较复杂和琐碎的部分，这里要处理各种各样的可能发生的情况。如果前面的二进制数字位数小于等于后面的位数，由于前一个数字在第一部分时已经全部被改写成了+，当读写头往左寻找加数时读到 B，就可以代表前面数字全部被加到后面的数字上。此时需要去被加完后得到的后一个数字那里查看，在数字 01（多出的长度未参与运算）和字母 LY（参与运算的后面位）之间有没有一个待进位的 2 出现。如果没有，后一个数字就是结果，进入打印阶段。如果有，分为两种情况：a）2 前面还有 0 或 1，直接向左进位后得到结果，进入打印阶段。b）2 前面就是+，那么将 2 改写为 0，然后将左边的+改写成 1 完成进位，得到结果，进入打印阶段。如果前面的二进制数字位数大于后面的，当后一个二进制数被全部改写为字母 LY 时，表示前面的位数多于后面的。当后面数字最前面位数改写完后，有两种情况。a）在准备返回的状态读到了左边的+，表示已经得到结果。b）在准备进位的状态读到了左边的+，那么就要一直越过所有的+向左，直至读到前一位二进制数的 01 部分（即没有参与运算、被改写为+的部分），然后开始向左进行进位操作。若是 0 改写为 1，若是 1 改写为 0 后再往左移动重复一次进位，至得到结果。然后进入打印阶段。

3. 打印阶段。这一部分比较简单，就是从左往右把等号左边的数字搬到后边打印出来就好。打印顺序是从左往右依次打印。0 和 1 搬到“=”号右边打印为 0 和 1，读取到“+”一律跳过不往“=”右边打印，L 和 Y 搬到“=”右边打印为 0 和 1。左边被读取过的 0，1，+，L，Y 一律改写为#。以上是整个实验的思路，最后添加头尾和细节就好，全部打印完后向右读取到“=”图灵机停机。

2 正确性与完备性证明

先进行一些测试，在证明的最后会详细说明该测试的目的。

实验 1：一进制加法

测试样例：111+11=11111

1+11111=111111

+=B

1+=1
+1=1
1111111+=1111111
实验 2：四位二进制加法（有前导零）
测试样例：0000+1111=01111
0001+1111=10000
1111+1111=11110
1010+0101=01111
1111+0001=10000
0000+0000=00000
实验 3：任意位二进制加法（无前导零）
10100111110111010+10101010111101=10111101001110111
10+10=100
111111000000+1000000=1000000000000
1+111111111111111=1000000000000000
0+0=0
0+11111=11111
10100+0=10100
111111111111111+1=1000000000000000
以上测试样例均通过。

在上面的所有测试数据中，我们使用了各种情况的数据来检验状态转移表的完备性和正确性。

实验一测试了不同长度的一进制数相加，也尝试了“+=”、“1+=”、“+1=”等合法但是边缘状态的输入，考虑到任何一种输入（甚至包括无限长的情况），我们都知道，它一定可以归为上面情况的其中一类（因为非边缘情况都是“不同长度一位数相加”这种情况）。那么我们成功遍历了所有可能出现的情况，以及实验代码的各个部分，全部通过。于是我们断言正确，且完备性成立。

实验二也是同样测试了许多一般的情况，也包括有无前导零、全不进位、全部进位、交错进位等特殊的情况，以及这些各种数据情况的可能排列组合，最终覆盖了所有输入情况和代码部分，全部通过。我们断言正确，且完备性成立。

实验三在实验二除去前导零的合法输入里，又加上了前后输入数字位数长短这个条件，将前远远短于后、前远远长于后、前后相等、前后只相差一位、相差两位等情况加入并排列组合来测试，全部通过。我们断言正确，且完备性成立。

值得注意的是，这里我利用了“特征输入”的思想来证明，也即将一类输入集用一个特征输入来代表，显然地，任何合法输入都可以归类为某个输入集，在图灵机的“无限”背景下，这个输入集在特征输入下是正确的，那么该输入一定是正确的，因为即便是无限的情况，我们也容易看出：输入对结果的影响是平凡的，这是一种隐性归纳。所以通过上述证明，我断言我的状态转移表是正确且完备的。

3 对图灵机的一些理解与思考

3.1 关于弱化图灵机的计算能力的思考与证明

图灵机为什么能够通过有限规则处理无限输入？原因在于图灵机的特性。图灵机是计算完备的，任何人类能够用纸笔进行的运算都能在图灵机上实现，这么强大的计算能力很大程度上依赖于理想图灵机的物理特征——无限空间、可自由移动的读写头等等，“状态”这个参量的引入和图灵机这些特性保证了每一个“循环”能够进行和停机，因为每一种状态能够容纳的信息是无限的。

因此，一旦这些物理特征受到限制，图灵机的计算能力很可能会下降，下面我们将思考并证明这些具体的影响。

1. 读写头只能读不能写。读写头写的能力是图灵机最重要的能力之一，一旦读写头不能够写，因为无法对纸带进行任何改变，图灵机将无法完成大部分计算，例如在某进制下的加法运算、排序操作等等，它几乎无法解决所有可计算可编程的问题。此时图灵机计算能力近乎于无。

2. 读写头只能向右移动。此时图灵机无法改写它已经读取过的符号，有些问题是无法解决的，例如交换两个符号的位置。而对于其它一些问题，比如实验二中的四位二进制加法，我们或许可以通过穷举并设置很多状态来解决它，但是一旦输入允许的符号集是无限的，我们就必然需要无限个状态来对应，这在代码上是无法操作的，因为我们不可能写无限行代码；然而，如果符号集有限，我们则一定可以通过设置足够多的状态，来完成计算。此时图灵机计算能力会比理想状态弱。

3. 字符集有限。我们首先断言一个事实：有限字符集中所有字符可以扩充成无限字符。因为我们可以通过若干个有限字符拼接而形成一个新的复合字符(例如我们只有 1、2 这两个字符，我们可以规定 22 在意义上对应 2，222 在意义上对应 3，2222 在意义上对应 4...并且把 1 作为分隔符号于是，类似于 ASCLL 码，我们得到了一个字符对应表)。这种情况下，图灵机计算能力不受影响。

但是，如果考虑广义符号（也就是说，不局限于已知的通用符号，还包括自创符号、象形符号等），我们断言图灵机计算能力是会下降的。如果我们只有 1 这个字符，我们无法通过上面的对应方法创建类似 ASCLL 码的对应表，因为自然数集合的势是$\aleph_0$。的，如果我们认为一个实数是一个字符，需要表示全体实数的话，将会面临 $\aleph_1$的难题，根据连续统原理，它是超越$\aleph_0$。的，无法用若干个 1 拼接起来表示，也就是说我们无法找到这样一个双射，也就无法创建这样一个对应表。在这种情况下图灵机的计算能力会降低。当然，这种情况是不存在的，因为图灵机的定义表明字符集不是无穷的。

4. 状态数有限。设状态数为 n，考虑下面这个计算问题：把“=”后面 n+1 个 B 改成 n+1 个不同的字符。我们断言这是无法完成的，证明如下：首先，一个状态下读 B 只能改成一个字符，这是显然的，因为如若不是这样，它将在读到 B 时无法进行下去，因为它无法判断究竟该改成哪个字符。那么我们就可以知道，

这 n 个状态最多只能将“=”后面的 B 改为 n 个不同字符。因此这种计算问题是无法被限制状态数的图灵机所解决的。此时图灵机计算能力降低。

5.纸带长度有限。结论是显而易见的。假设纸带长度为 n，我们无法计算一进制加法“111...11（n 个 1）+1=”。一般地，考虑到当输入足够长（比纸带长度要长）的时候，图灵机甚至无法成功进行输入操作，自然也就无法完成计算任务。图灵机的计算能力下降。

3.2 关于比图灵机更强计算模型的思考

图灵机计算能力已经非常强大，可是世界上究竟还存不存在比图灵机更加强大的模型？我认为结论应该是不证自明的：有。因为这正是计算机科学的生命力所在。下面是我结合相关资料对更强大的计算模型（超计算模型）的思考。

1. 谕示机。这是带“黑箱”的图灵机，当年由阿兰·图灵本人提出，“黑箱”就是一个谕示，经过这样的谕示可以直接得到结果。一部预言机可以视为是与一个预言者（oracle）相连接的图灵机。所谓预言者的概念，是一个可以回答特定问题集合的一个实体，而且常常使用特定的自然数子集 A 来表示这个问题。我们可以很自然的发现，一部预言机可以执行很多对一般图灵机来说很特殊的操作，并且可以借由询问预言者来获得"x 是否在 A 内?"这种特定形式问题的解答。一部预言机，基本上必定包含一整个图灵机。在我的观点中，这样一种谕示机仍然难以在现实中实现，因为它甚至比理想图灵机还要理想，这样一种“谕示集”绝非有限所能涵盖，其谕示规则又需要特殊规定。因此，尽管谕示机很强，它是极难实现的。

2.量子计算机。在量子物理中，对一个系统状态的完整描述需要使用复数，即量子系统的状态是一个位于向量空间中的一个点。右括向量 $|x/rangle+$意味着 x 是一个(纯)量子态。与这个量子系统相关的希尔伯特空间是复向量（complex vector space）空间，系统在任何时候的状态由这个希尔伯特空间中的一个单位长度向量表示。

同时系统的叠加态可表示为：$/sum_{i=0}^{2^n-1} a_i|S_i/rangle+$，其中振幅 a_i 为满足 $/Sigma +_i |a_i|^2 = 1$的复数。

为了使用物理系统进行计算，我们必须能够改变系统的状态。量子力学定律只允许状态向量的幺正转换。一个幺正矩阵的共轭转置等于它的逆矩阵，并且要求状态变换由酉矩阵来表示，这就保证了得到所有可能结果的概率之和为 1。同时量子电路(和量子计算机)的定义只允许局部幺正转换；也就是说，对固定数目的比特进行酉（幺正）变换。

值得注意的是：量子计算机的计算能力在本质上与图灵机等价，但在计算复杂度上可以优于图灵机（如果这也算是计算能力的话。）。现实中的量子计算机的计算能力可以在多项式时间内解决 BQP ，并没有想象中的那么强。

但是，尽管目前可以通过结构与算法优化使计算能力不断提高，但量子计算机的计算能力还是有真正的上限的：即布莱梅曼极限（Bremermann's limit）。在量子物理框架下，我们宇宙中所有物质的计算能力都不可能超过每千克 c^2/h bits/s（h 为普朗克常数，c 为光速）这也是量子计算机真正无法逾越的计算速度极限。而且我们也不可能真正地达到该极限，因为所需能量会使计算机直接坍塌成一个黑洞。

最后值得一提的是，只要对量子力学中算符的线性要求做些微的放宽，例如，温伯格引入的非线性算符（这些工作出现在温伯格试图研讨的所谓非线性量子力学中）得到允许，则我们可以在新型量子计算机上用多项式时间求解 PSPACE 完全问题（ NP 完全问题也自然不在话下）。但是由于非线性的引入一定会同时容许超光速通信和违背热力学第二定律的结果，所以提议基本是不可行的。

3.相对论效应。这也许是最异想天开的超越图灵机计算能力的方式了。在相对论中，不同物体参考系的时间流逝不一样，如果我们能让计算机参考系在时间流逝上快很多，那我们也变相得解决了这个问题。一台计算机留在地球上让它做一个复杂的计算问题，然后操作者登上一个航天器，加速到接近光速，一段时间后减速再返回地球即可。只是，这种方法虽然脑洞大开，看似完美地利用了时间法则，却绕不开一个问题:人类距离光速还极其遥远，甚至接近光速所需要的能量用来进行计算都会比直接利用相对论效应效率要高。

3.3 图灵机背后的数理思想畅论

图灵机如此强大且形式简洁，其背后蕴含的数理思想也是值得我们探究的。刚学习图灵机时，看到“状态转移图”，想到曾经学过的“马尔科夫链”，也是通过一定规则下的状态转移来实现建模与计算的，与此同时，这种“状态转移”类似于人脑的计算流程，是一种很好的类脑实验。

不仅如此，较为复杂的图灵机模型还包含了“循环”思想，也就是首尾圆合、逻辑自洽，这体现在实验过程中就是三个实验中的各个循环（计算阶段循环、进位阶段循环等），而自洽的要求则体现在每个循环的边缘时刻都必须畅通无阻地过渡，因此不难看出，其状态转移背后的“链”是处处光滑的。