图灵机实验报告

刘　迅

2022 年 4 月 21 日

目录

1 图灵机设计

1.1 一进制加法图灵机

1.1.1 核心想法

在等号左每读一位 1，便向等号右加一位 1.

重复这一过程，等号右于是有等号左式的 1 的总数，即左式的一进制加法.

1.1.2 状态转移图

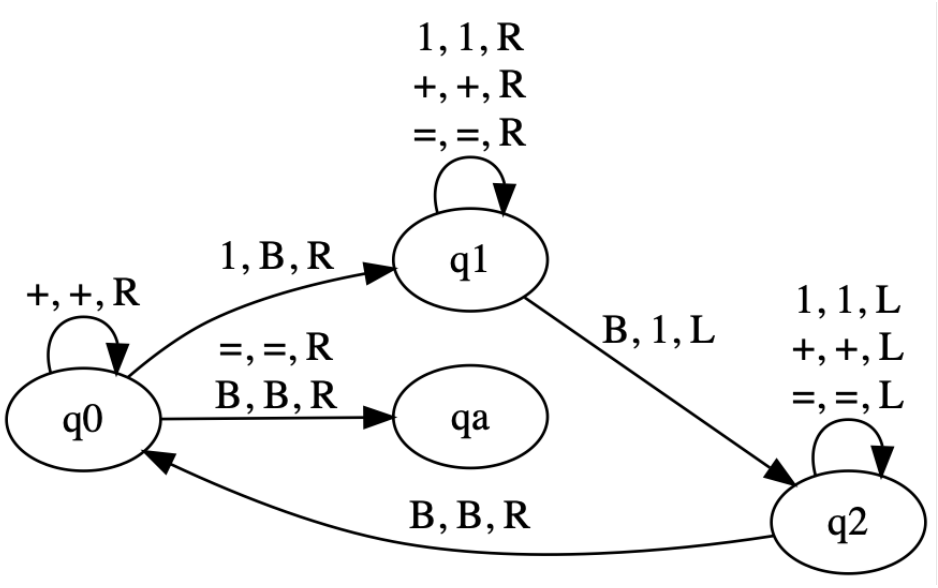


图 1: 实现一进制加法的图灵机

1.1.3 状态转移表

限于篇幅，展示于附录 4.1 中（有超链接可以点击 4.1）。

1.1.4 设计细节

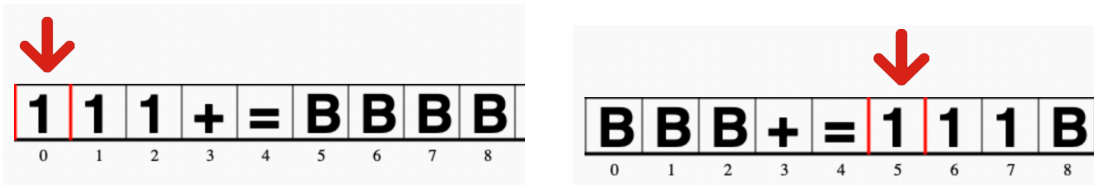
q0 为初始状态，q1 为向右扩张状态，q2 为向左收缩状态，qa 为接收状态。

考虑从左端读取一个 1，首先擦除这个 1，接下来要向右找到第一个 B 位置改写成 1，然后再向左找到第一个位置 1，之后重复开头的过程。接受状态是等号左侧的 1 读完了，表现为 q0 状态读到 = 或者 B 字符。

关于状态转移图，非自环边的转移如上所述，自环边的含义是一直寻找转移所需要达到的合法状态。

另外考察此程序的边界情况：

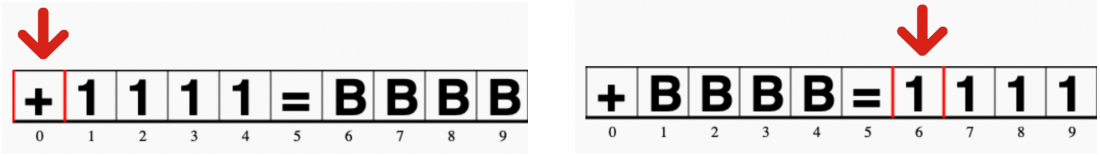
情形一： + 左为空



(a) 输入 (b) 输出

图 2: + 左为空

情形二： + 右为空



(a) 输入 (b) 输出

图 3: + 右为空

1.2 4 位二进制加法图灵机

1.2.1 核心想法

首先把把等号左边两个四位数不进位按位相加，把得到的五位数放到等号右边。例如 1001+0101=01102.

然后考虑处理右式的进位，即得到答案。

1.2.2 状态转移图

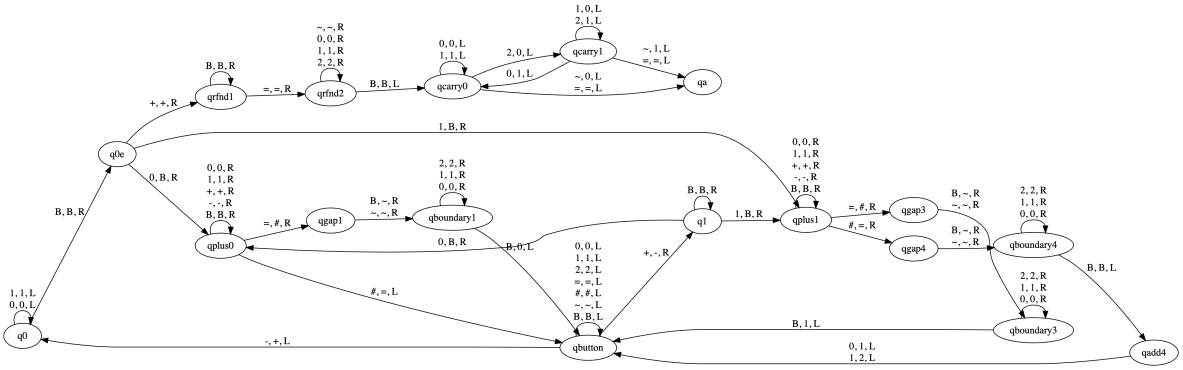


图 4: 实现固定 4 位二进制加法的图灵机

1.2.3 状态转移表

限于篇幅，展示于附录 4.2 中（有超链接可以点击 4.2）。

1.2.4 设计细节

为了节省相同状态，上述设计在左式两数相加的时候翻转了 + 和 =，用来标记当前应该处理的是第一个加法元还是第二个加法元。

加法部分： q0 状态是寻找第一个加法元的首位，q1 状态是寻找第二个加法元的首位；qplus 状态处理对等号右边的加法；qgap 是为了把等号左边的四位数变成等号右边的五位数，因此需要在等号右第一个位置留空；qboundary 状态是处理加法时向右寻找最低位；qbutton 状态是根据翻转的 + 或 − 判断接下来处理第一个加法元还是第二个加法元，如果当前是 +，则向右处理 q1，如果当前是 −，则向左处理 q0。

加法的边界条件： q0 向左找到第一个 B 时，转向右寻找第一个加法元的首位，如果此时读到 +，意味着第一个加法元已经读完，也就是等号左边的四位数都处理完了，于是转到进位的处理。

进位部分： 首先依靠 qrfnd1 状态把读写头移到等号位置，qrfnd2 状态把读写头移到右式最低位；qcarry 状态分别代表进位 0,1,2。以 qcarry 状态从低到高位处理进位：

qcarry0 遇 2 写 0 并进入 qcarry1，qcarry1 遇 1 写 0 并进入 qcarry0；qcarry1 遇 2 写 1 并进入 qcarry1。边界条件是读取到预先留空的第五位，填充上当前所剩进位后，进入 qa 状态结束图灵机。

考察尚未进位的中间状态，

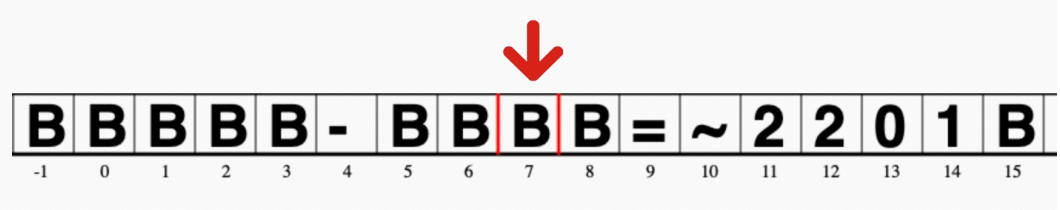


图 5: 进位：以 1101+1100 为例

另考察两组边界情况：

情形一： 1111+1111=

(a) 输入 (b) 输出

图 6: 1111+1111=

情形二： 0000+0000=

(a) 输入 (b) 输出

图 7: 0000+0000=

1.3 任意位二进制加法图灵机

1.3.1 核心想法

首先把两个加法元按最低位到最高位不进位相加，依次放在等号右边。例如1011+1=2101，注意到得到的结果和正确答案相比，其一是没有进位，其二是逆序的。

对逆序的结果进位，沿用上例，得到 0011.

再对进位后的结果翻转，沿用上例，得到 1100，即为正确结果.

1.3.2 状态转移图

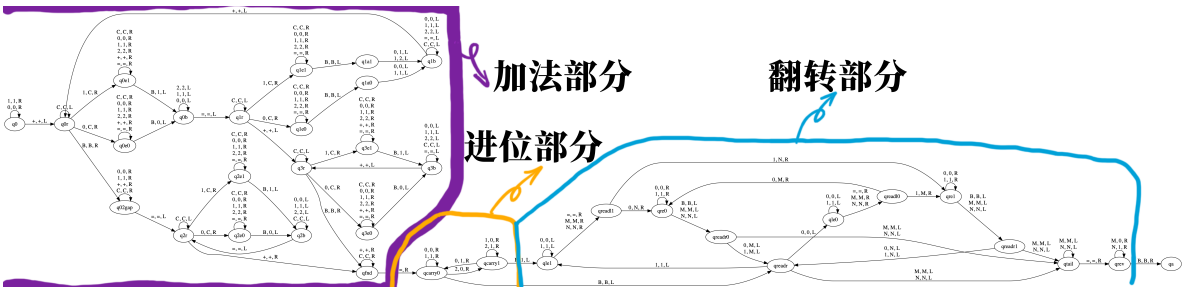


图 8: 实现任意位二进制加法的图灵机

1.3.3 状态转移表

限于篇幅，展示于附录 4.3 中（有超链接可以点击 4.3）。

1.3.4 设计细节

粗粒度的流程图如图9所示。

关于做法中的“从低到高位相加，结果放在等式右边”与“二进制加法进位”操作，之前已经阐述过，这里就不再重复。接下来说明做法中最后一步的翻转操作。

翻转操作： 用带字符 M 表示处理过的 0，N 表示处理过的 1. 首先将读写头移动到式子最右端的数字。按照 0 或 1 分两类情况，分别进入状态 qle0 与 qle1，将最左端的数字擦除为 M 或 N. 按式子最左端的数字分 0 或 1 两类情况，分别进入状态

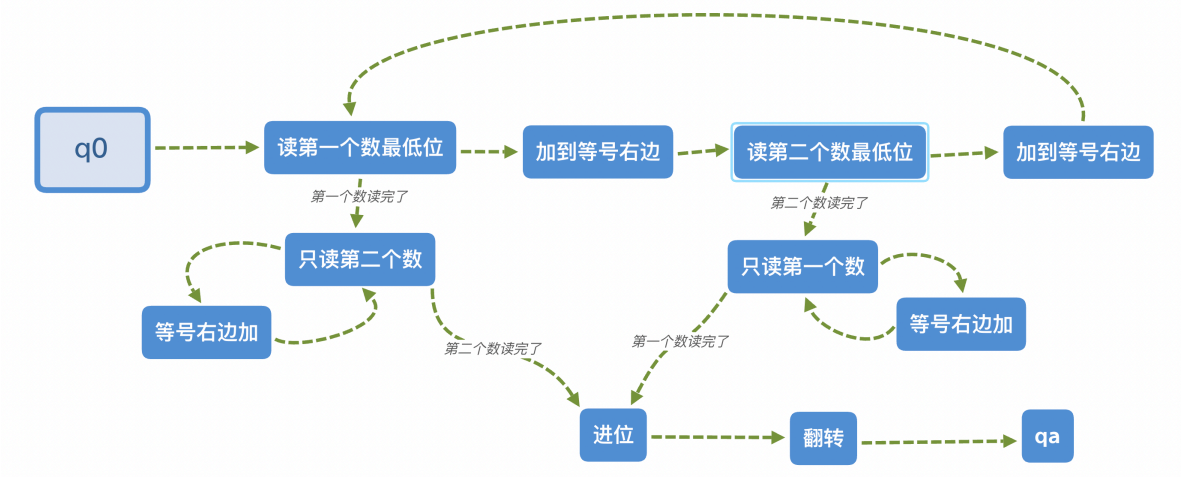


图 9: 粗粒度流程图：任意位二进制加法图灵机

qre0 与 qre1，将最右端的数字擦除为 M 或 N. 重复以上过程，直到 0,1 被全部读完。最后将所有的 M 与 N 复原为 0 与 1，即得到正确答案。

考察翻转的中间状态,

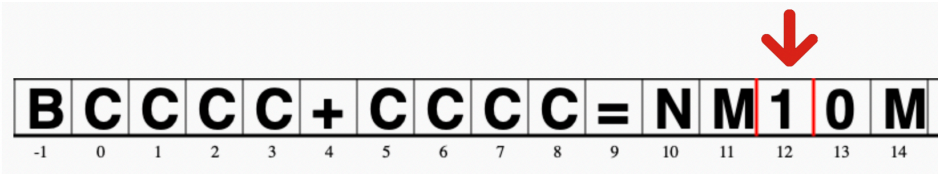
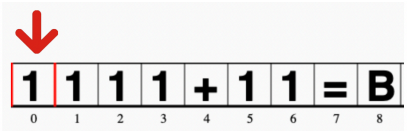


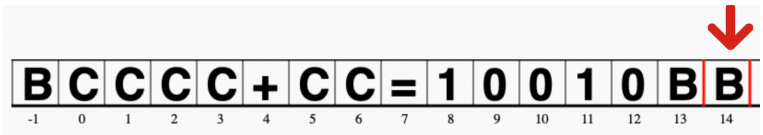
图 10: 翻转：以 1100+1000 为例

另考察边界情况:

情形： 1111+11=



(a) 输入



(b) 输出

图 11: 1111+11=

2 正确性证明

三个问题的图灵机设计都是模仿人类解决问题的思路，分步骤设计。对于一般情形的正确性类比于人类处理进制加法问题的正确性。

各题的边界情况，已经在“图灵机设计”部分的1.1.4,1.2.4与1.3.4中考察。

三个设计均通过了教学平台的黑箱测试，得分为满分。

3 对图灵机的理解

3.1 有限规则处理无限输入

图灵机状态转移图中的环结构，相当于算法设计中的循环结构。因而对于任意长度的输入，都可以用有限的规则处理。

3.2 带字符集与信息编码

扩充带字符集的行为，本质上是利用带字符对图灵机所处状态进行编码，从而减少状态转移图中状态的规模。

类比一下，选择扩大带字符集来设计图灵机、而不是多设立状态来设计图灵机，就像是用高级语言编程，而不是把每一个语句都写成汇编甚至二进制的机器码。

由此看来，设计图灵机时选择扩充带字符集是值得提倡的，这样可以减少状态设计的规模，方便设计。

3.3 通过读写带字符实现条件判断

承接上一条对带字符的思考。每一个时刻，图灵机能得到的信息只有当前所处图灵机的状态节点、纸带的位置、当前纸带位置的字符与读写头的方向。

为了实现条件判断，可以通过对纸带上字符的读写。例如“4 位二进制加法图灵机”的设计中，每次读到“+”时翻转为“-”，读到“-”时翻转为“+”，这样就实现了图灵机对左右两个加法元的交替读写。

3.4 带字符集大小 / 纸带长度 / 图灵机状态节点数量受到限制

从信息编码的观点看，无论是带字符集、纸带长度还是图灵机的状态节点数量受到限制，都是从一个维度上限制了信息编码的能力。如果其他的维度的信息编码能力仍然是无限的，那么整个图灵机信息编码的能力不会受到影响，也就是可解决的问题规模不会受到影响；而如果各个维度上信息编码的能力都受到了限制，那么图灵机的计算能力就会被削弱。

4 附录 A 三种图灵机的状态转移表

4.1 一进制加法图灵机

当前状态	读取	写入	转移头方向	下个状态
q0	1	B	→	q1
q0	+	+	→	q0
q0	=	=	→	qa
q0	B	B	→	qa
q1	1	1	→	q1
q1	+	+	→	q1
q1	=	=	→	q1
q1	B	1	←	q2
q2	1	1	←	q2
q2	+	+	←	q2
q2	=	=	←	q2
q2	B	B	→	q0

表 1: 一进制加法图灵机状态转移表

4.2 4 位二进制加法图灵机

当前状态	读取	写入	转移头方向	下个状态
q0	1	1	←	q0
q0	0	0	←	q0
q0	B	B	→	q0e
q0e	0	B	→	qplus0
q0e	1	B	→	qplus1
qplus0	0	0	→	qplus0
qplus0	1	1	→	qplus0
qplus0	+	+	→	qplus0

qplus0	-	-	→	qplus0
qplus0	B	B	→	qplus0
qplus0	=	#	→	qgap1
qplus0	#	=	←	qbutton
qgap1	B	∼	→	qboundary1
qgap1	∼	∼	→	qboundary1
qboundary1	2	2	→	qboundary1
qboundary1	1	1	→	qboundary1
qboundary1	0	0	→	qboundary1
qboundary1	B	0	←	qbutton
qplus1	0	0	→	qplus1
qplus1	1	1	→	qplus1
qplus1	+	+	→	qplus1
qplus1	-	-	→	qplus1
qplus1	B	B	→	qplus1
qplus1	=	#	→	qgap3
qplus1	#	=	→	qgap4
qgap3	B	∼	→	qboundary3
qgap3	∼	∼	→	qboundary3
qgap4	B	∼	→	qboundary4
qgap4	∼	∼	→	qboundary4
qboundary3	2	2	→	qboundary3
qboundary3	1	1	→	qboundary3
qboundary3	0	0	→	qboundary3
qboundary3	B	1	←	qbutton
qboundary4	2	2	→	qboundary4
qboundary4	1	1	→	qboundary4
qboundary4	0	0	→	qboundary4
qboundary4	B	B	←	qadd4
qadd4	0	1	←	qbutton
qadd4	1	2	←	qbutton

qbutton	0	0	←	qbutton
qbutton	1	1	←	qbutton
qbutton	2	2	←	qbutton
qbutton	=	=	←	qbutton
qbutton	#	#	←	qbutton
qbutton	∼	∼	←	qbutton
qbutton	B	B	←	qbutton
qbutton	-	+	←	q0
qbutton	+	-	→	q1
q1	B	B	→	q1
q1	0	B	→	qplus0
q1	1	B	→	qplus1
q0e	+	+	→	qrfnd1
qrfnd1	B	B	→	qrfnd1
qrfnd1	=	=	→	qrfnd2
qrfnd2	∼	∼	→	qrfnd2
qrfnd2	0	0	→	qrfnd2
qrfnd2	1	1	→	qrfnd2
qrfnd2	2	2	→	qrfnd2
qrfnd2	B	B	←	qcarry0
qcarry0	0	0	←	qcarry0
qcarry0	1	1	←	qcarry0
qcarry0	2	0	←	qcarry1
qcarry1	0	1	←	qcarry0
qcarry1	1	0	←	qcarry1
qcarry1	2	1	←	qcarry1
qcarry0	∼	0	←	qa
qcarry1	∼	1	←	qa
qcarry0	=	=	←	qa
qcarry1	=	=	←	qa

表 2: 4 位二进制加法图灵机状态转移表

4.3 任意位二进制加法图灵机

当前状态	读取	写入	转移头方向	下个状态
q0	1	1	→	q0
q0	0	0	→	q0
q0	+	+	←	q0r
q0r	C	C	←	q0r
q0r	0	C	→	q0e0
q0r	1	C	→	q0e1
q0e0	C	C	→	q0e0
q0e0	0	0	→	q0e0
q0e0	1	1	→	q0e0
q0e0	2	2	→	q0e0
q0e0	+	+	→	q0e0
q0e0	=	=	→	q0e0
q0e0	B	0	←	q0b
q0e1	C	C	→	q0e1
q0e1	0	0	→	q0e1
q0e1	1	1	→	q0e1
q0e1	2	2	→	q0e1
q0e1	+	+	→	q0e1
q0e1	=	=	→	q0e1
q0e1	B	1	←	q0b
q0b	2	2	←	q0b
q0b	1	1	←	q0b
q0b	0	0	←	q0b
q0b	=	=	←	q1r
q1r	C	C	←	q1r
q1r	0	C	→	q1e0
q1r	1	C	→	q1e1
q1e0	C	C	→	q1e0

q1e0	0	0	→	q1e0
q1e0	1	1	→	q1e0
q1e0	2	2	→	q1e0
q1e0	=	=	→	q1e0
q1e0	B	B	←	q1a0
q1a0	0	0	←	q1b
q1a0	1	1	←	q1b
q1e1	C	C	→	q1e1
q1e1	0	0	→	q1e1
q1e1	1	1	→	q1e1
q1e1	2	2	→	q1e1
q1e1	=	=	→	q1e1
q1e1	B	B	←	q1a1
q1a1	0	1	←	q1b
q1a1	1	2	←	q1b
q1b	0	0	←	q1b
q1b	1	1	←	q1b
q1b	2	2	←	q1b
q1b	=	=	←	q1b
q1b	C	C	←	q1b
q1b	+	+	←	q0r
q0r	B	B	→	q02gap
q02gap	0	0	→	q02gap
q02gap	1	1	→	q02gap
q02gap	+	+	→	q02gap
q02gap	C	C	→	q02gap
q02gap	=	=	←	q2r
q2r	C	C	←	q2r
q2r	0	C	→	q2e0
q2r	1	C	→	q2e1
q2e0	C	C	→	q2e0

q2e0	0	0	→	q2e0
q2e0	1	1	→	q2e0
q2e0	2	2	→	q2e0
q2e0	=	=	→	q2e0
q2e0	B	0	←	q2b
q2e1	C	C	→	q2e1
q2e1	0	0	→	q2e1
q2e1	1	1	→	q2e1
q2e1	2	2	→	q2e1
q2e1	=	=	→	q2e1
q2e1	B	1	←	q2b
q2b	0	0	←	q2b
q2b	1	1	←	q2b
q2b	2	2	←	q2b
q2b	C	C	←	q2b
q2b	=	=	←	q2r
q1r	+	+	←	q3r
q3r	C	C	←	q3r
q3r	0	C	→	q3e0
q3r	1	C	→	q3e1
q3e0	C	C	→	q3e0
q3e0	0	0	→	q3e0
q3e0	1	1	→	q3e0
q3e0	2	2	→	q3e0
q3e0	+	+	→	q3e0
q3e0	=	=	→	q3e0
q3e0	B	0	←	q3b
q3e1	C	C	→	q3e1
q3e1	0	0	→	q3e1
q3e1	1	1	→	q3e1
q3e1	2	2	→	q3e1

q3e1	+	+	→	q3e1
q3e1	=	=	→	q3e1
q3e1	B	1	←	q3b
q3b	0	0	←	q3b
q3b	1	1	←	q3b
q3b	2	2	←	q3b
q3b	C	C	←	q3b
q3b	=	=	←	q3b
q3b	+	+	←	q3r
q2r	+	+	→	qfnd
q3r	B	B	→	qfnd
qfnd	+	+	→	qfnd
qfnd	C	C	→	qfnd
qfnd	=	=	→	qcarry0
qcarry0	0	0	→	qcarry0
qcarry0	1	1	→	qcarry0
qcarry0	2	0	→	qcarry1
qcarry1	0	1	→	qcarry0
qcarry1	1	0	→	qcarry1
qcarry1	2	1	→	qcarry1
qcarry0	B	B	←	qreadr
qreadr	0	0	←	qle0
qreadr	1	1	←	qle1
qcarry1	B	1	←	qle1
qle1	0	0	←	qle1
qle1	1	1	←	qle1
qle1	=	=	→	qreadl1
qle1	M	M	→	qreadl1
qle1	N	N	→	qreadl1
qreadl1	0	N	→	qre0
qreadl1	1	N	→	qre1

qle0	0	0	←	qle0
qle0	1	1	←	qle0
qle0	=	=	→	qreadl0
qle0	M	M	→	qreadl0
qle0	N	N	→	qreadl0
qreadl0	0	M	→	qre0
qreadl0	1	M	→	qre1
qre0	0	0	→	qre0
qre0	1	1	→	qre0
qre1	0	0	→	qre1
qre1	1	1	→	qre1
qre0	B	B	←	qreadr0
qre0	M	M	←	qreadr0
qre0	N	N	←	qreadr0
qre1	B	B	←	qreadr1
qre1	M	M	←	qreadr1
qre1	N	N	←	qreadr1
qreadr0	0	M	←	qreadr
qreadr0	1	M	←	qreadr
qreadr1	0	N	←	qreadr
qreadr1	1	N	←	qreadr
qreadr0	M	M	←	qtail
qreadr0	N	N	←	qtail
qreadr1	M	M	←	qtail
qreadr1	N	N	←	qtail
qreadr	M	M	←	qtail
qreadr	N	N	←	qtail
qtail	M	M	←	qtail
qtail	N	N	←	qtail
qtail	=	=	→	qrev
qrev	M	0	→	qrev

qrev	N	1	→	qrev
qrev	B	B	→	qa

表 3: 任意位二进制加法图灵机状态转移表