图灵机

图灵机

- **图灵机(Turing machine)**是由图灵**(Alan Mathisom Turing)**在**1936**年提出的，它是一个通用的计算模型。
- 通过研究**TM**，来研究**递归可枚举集(recursively enumerable set)**和**部分递归函数(partial recursive function)**。
- 对算法和可计算性研究提供形式化描述工具。

图灵机

- **有效过程(effective procedure)与算法(algorithm)。**
- 具有有穷描述的过程是可数无穷多的，但函数却是不可数无穷多的。
- 世界上存在着许多的问题和函数，是无法用具有有穷描述的过程完成计算的——**是不可计算的(incomputable)** 。

图灵机

- 主要内容
  - **TM作为一个计算模型，它的基本定义，即时描述，TM接受的语言；TM的构造技术；TM的变形；Church-Turing论题；通用TM。可计算性、P-NP问题。**
- 重点
  - **TM的定义、TM的构造。**
- 难点
  - **TM的构造。**

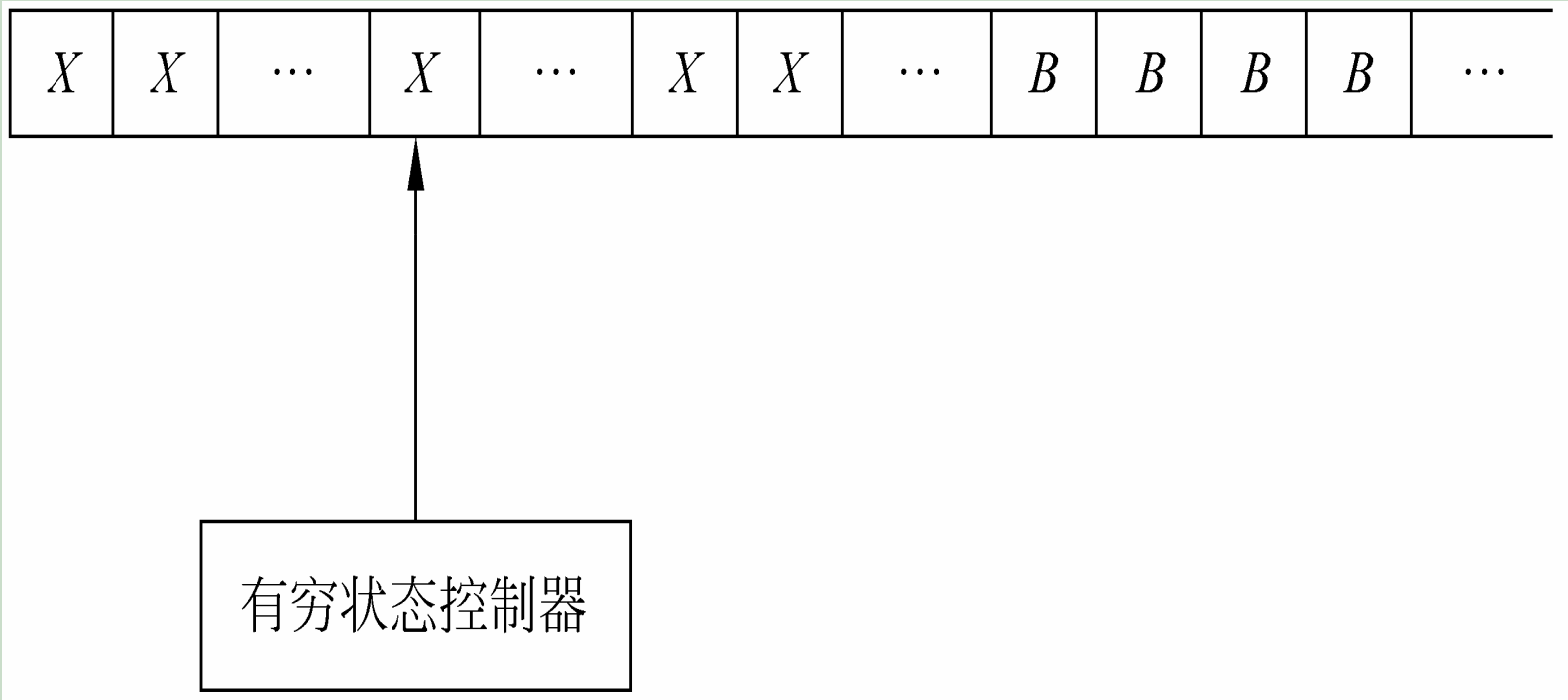
基本概念

- 图灵提出TM具有以下两个性质
  - 具有有穷描述。
  - 过程必须是由离散的、可以机械执行的步骤组成。
- 基本模型包括
  - 一个有穷控制器。
  - 一条含有无穷多个带方格的输入带。
  - 一个读头。

基本概念

- 一个移动将完成以下三个动作：
  - 改变有穷控制器的状态；
  - 在当前所读符号所在的带方格中印刷一个符号；
  - 将读头向右或者向左移一格。

直观物理模型



基本TM

- 图灵机(Turing machine)/基本的图灵机

$$TM\ M=(Q, \Sigma, \Gamma, \delta, q_0, B, F),$$

- Q为状态的有穷集合，$\forall q \in Q$，q为M的一个状态；
- $q_0 \in Q$，是M的开始状态，对于一个给定的输入串，M从状态q_0启动，读头正注视着输入带最左端的符号；

基本TM

- **$F \subseteq Q$，是M的终止状态集，$\forall q \in F$，q为M的一个终止状态。与FA和PDA不同，一般地，一旦M进入终止状态，它就停止运行；**

- **Γ 为带符号表(tape symbol)，$\forall X \in \Gamma$，X为M的一个带符号，表示在M的运行过程中，X可以在某一时刻出现在输入带上；**

基本TM

- **B∈Γ，被称为空白符(blank symbol)，含有空白符的带方格被认为是空的；**
- **Σ⊆Γ-{B}为输入字母表，∀a∈Σ，a为M的一个输入符号。除了空白符号B之外，只有Σ中的符号才能在M启动时出现在输入带上；**

基本TM

- δ：Q×Γ→Q×Γ×{R, L}，为M的移动函数(transaction function)。
- δ(q , X)=(p , Y, R)表示M在状态q读入符号X，将状态改为p，并在这个X所在的带方格中印刷符号Y，然后将读头向右移一格；
- δ(q , X)=(p , Y , L)表示M在状态q读入符号X，将状态改为p，并在这个X所在的带方格中印刷符号Y，然后将读头向左移一格。

基本TM

- 例 设$M_1=(\{q_0, q_1, q_2\},\{0, 1\},\{0, 1, B\}, \delta, q_0, B, \{q_2\})$，其中 δ 的定义如下，对于此定义，也可以用表格表示。

$$\delta(q_0, 0)= (q_0, 0, R)$$

$$\delta(q_0, 1)= (q_1, 1, R)$$

$$\delta(q_1, 0)= (q_1, 0, R)$$

$$\delta(q_1, B)= (q_2, B, R)$$

基本TM

	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	
q_1	$(q_1, 0, R)$		(q_2, B, R)
q_2			

基本TM

- 即时描述(instantaneous description, ID)
- $\alpha_1\alpha_2 \in \Gamma^*$，$q \in Q$，$\alpha_1 q \alpha_2$称为M的即时描述
  - q为M的当前状态。
  - $\alpha_1\alpha_2$为M的输入带最左端到最右的非空白符号组成的符号串或者是M的输入带最左端到M的读头注视的带方格中的符号组成的符号串
  - M正注视着α_2的最左符号。

基本TM

设$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$是M的一个ID

- 如果$\delta(q, X_i)=(p, Y, R)$，则M的下一个ID为$X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$

  记作

  $$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash_M X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$$

  表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$下，经过一次移动，将ID变成$X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$。

基本TM

- 如果 $\delta(q, X_i)=(p, Y, L)$则

  当$i \neq 1$时，M的下一个ID为

  $$X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$$

  记作

  $$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash_M X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$$

  表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$下，经过一次移动，将ID变成$X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$;

基本TM

- $\vdash_M$是$\Gamma^*Q\Gamma^* \times \Gamma^*Q\Gamma^*$上的一个二元关系
  - $\vdash_M^n$表示$\vdash_M$的n次幂：$\vdash_M^n = (\vdash_M)^n$
  - $\vdash_M^+$表示$\vdash_M$的正闭包：$\vdash_M^+ = (\vdash_M)^+$
  - $\vdash_M^*$表示$\vdash_M$的克林闭包：$\vdash_M^* = (\vdash_M)^*$
- 在意义明确时，分别用$\vdash$、$\vdash^n$、$\vdash^+$、$\vdash^*$表示$\vdash_M$、$\vdash_M^n$、$\vdash_M^+$、$\vdash_M^*$。

基本TM

- 例 上例所给的M_1在处理输入串的过程中经历的ID变换序列。

（1）处理输入串000100的过程中经历的ID的变换序列如下：

$$q_0000100 \vdash_M 0q_000100 \vdash_M 00q_00100 \vdash_M 000q_0100$$
$$\vdash_M 0001q_100 \vdash_M 00010q_10 \vdash_M 000100q_1 \vdash_M 000100Bq_2$$

基本TM

（2）处理输入串0001的过程中经历的ID变换序列如下：

$q_0 0001 \vdash_M 0q_0 001 \vdash_M 00q_0 01$

$\vdash_M 000q_0 1 \vdash_M 0001q_1 \vdash_M 0001Bq_2$

（3）处理输入串000101的过程中经历的ID变换序列如下：

$q_0 000101 \vdash_M 0q_0 00101 \vdash_M 00q_0 0101$

$\vdash_M 000q_0 101 \vdash_M 0001q_1 01 \vdash_M 00010q_1 1$

基本TM

（4）处理输入串1的过程中经历的ID变换序列如下：

$q_0 1 \vdash_M 1q_1 \vdash_M 1Bq_2$

（5）处理输入串00000的过程中经历的ID变换序列如下：

$q_0 00000 \vdash_M 0q_0 0000 \vdash_M 00q_0 000$

$\vdash_M 000q_0 00 \vdash_M 0000\ q_0 0 \vdash_M 00000q_0 B$

基本TM

- TM接受的语言

  $L(M)=\{x \mid x\in\Sigma^* \ \&\ q_0x \vdash_M^* \alpha_1 q \alpha_2 \ \&\ q\in F \ \&\ \alpha_1、\ \alpha_2\in\Gamma^*\}$

- TM接受的语言叫做递归可枚举语言(recursively enumerable language，r.e.)。
- 如果存在TM $M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$，$L=L(M)$，并且对每一个输入串x，M都停机，则称L为递归语言(recursively language)。

基本TM

- 例 设有$M_2=(\{q_0, q_1, q_2, q_3\},\{0, 1\},\{0, 1, B\}, \delta, q_0, B, \{q_3\})$，其中$\delta$的定义如下：

$\delta(q_0, 0)= (q_0, 0, R)$

$\delta(q_0, 1)= (q_1, 1, R)$

$\delta(q_1, 0)= (q_1, 0, R)$

$\delta(q_1, 1)= (q_2, 1, R)$

$\delta(q_2, 0)= (q_2, 0, R)$

$\delta(q_2, 1)= (q_3, 1, R)$

基本TM

	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	
q_2	$(q_2, 0, R)$	$(q_3, 1, R)$	
q_3			

基本TM

- 为了弄清楚M_2接受的语言，需要分析它的工作过程。

（1）处理输入串00010101的过程中经历的ID变换序列如下：

$q_000010101 \vdash 0q_00010101 \vdash 00q_0010101$
$\vdash 000q_010101 \vdash 0001q_10101 \vdash 00010q_1101$
$\vdash 000101\ q_201 \vdash 000101\ 0\ q_21 \vdash 00010101q_3$

基本TM

- M_2在q_0状态下，遇到0时状态仍然保持为q_0，同时将读头向右移动一格而指向下一个符号；
- 在q_1状态下遇到第一个1时状态改为q_1，并继续右移读头，以寻找下一个1；在遇到第二个1时，动作类似，只是将状态改为q_2；当遇到第三个1时，进入终止状态q_3，此时它正好扫描完整个输入符号串，表示符号串被M_2接受。

基本TM

（2）处理输入串**1001100101100**的过程中经历的ID变换序列如下：

$q_0$1001100101100 ⊢ 1$q_1$001100101100

⊢ 10 $q_1$01100101100 ⊢ 100$q_1$1100101100

⊢ 1001 $q_2$100101100 ⊢10011$q_3$00101100

- M_2遇到第三个1时，进入终止状态q_3，输入串的后缀00101100还没有被处理。但是，由于M_2已经进入终止状态，表示符号串1001100101100被M_2接受

基本TM

（3）处理输入串000101000的过程中经历的ID变换序列如下：

$q_0000101000 \vdash 0q_000101000 \vdash 00q_00101000$

$\vdash 000q_0101000 \vdash 0001q_101000 \vdash 00010q_11000$

$\vdash 000101q_2000 \vdash 0001010\ q_200 \vdash 00010100\ q_20$

$\vdash 000101000\ q_2B$

- 当M_2的ID变为$000101000q_2B$时，因为无法进行下一个移动而停机,不接受输入串000101000。

基本TM

- M_2接受的语言是字母表{0，1}上那些至少含有3个1的0、1符号串。请考虑，如何构造出接受字母表{0，1}上那些含且恰含有3个1的符号串的TM。

基本TM

- 例 构造TM M_3，使$L(M)=\{0^n1^n2^n \mid n\geqslant 1\}$。

  分析：
  - 不能通过“数”0、1、或者2的个数来实现检查。
  - 最为原始的方法来比较它们的个数是否是相同的：消除一个0、然后消除一个1，最后消除一个2。
  - 消除的0的带方格上印刷一个X，在消除的1的带方格上印刷一个Y，在消除的2的带方格上印刷一个Z。

基本TM

- 正常情况下，输入带上的符号串的一般形式为

  00…0011…1122…22

- **TM**启动后，经过一段运行，输入带上的符号串的一般情况为

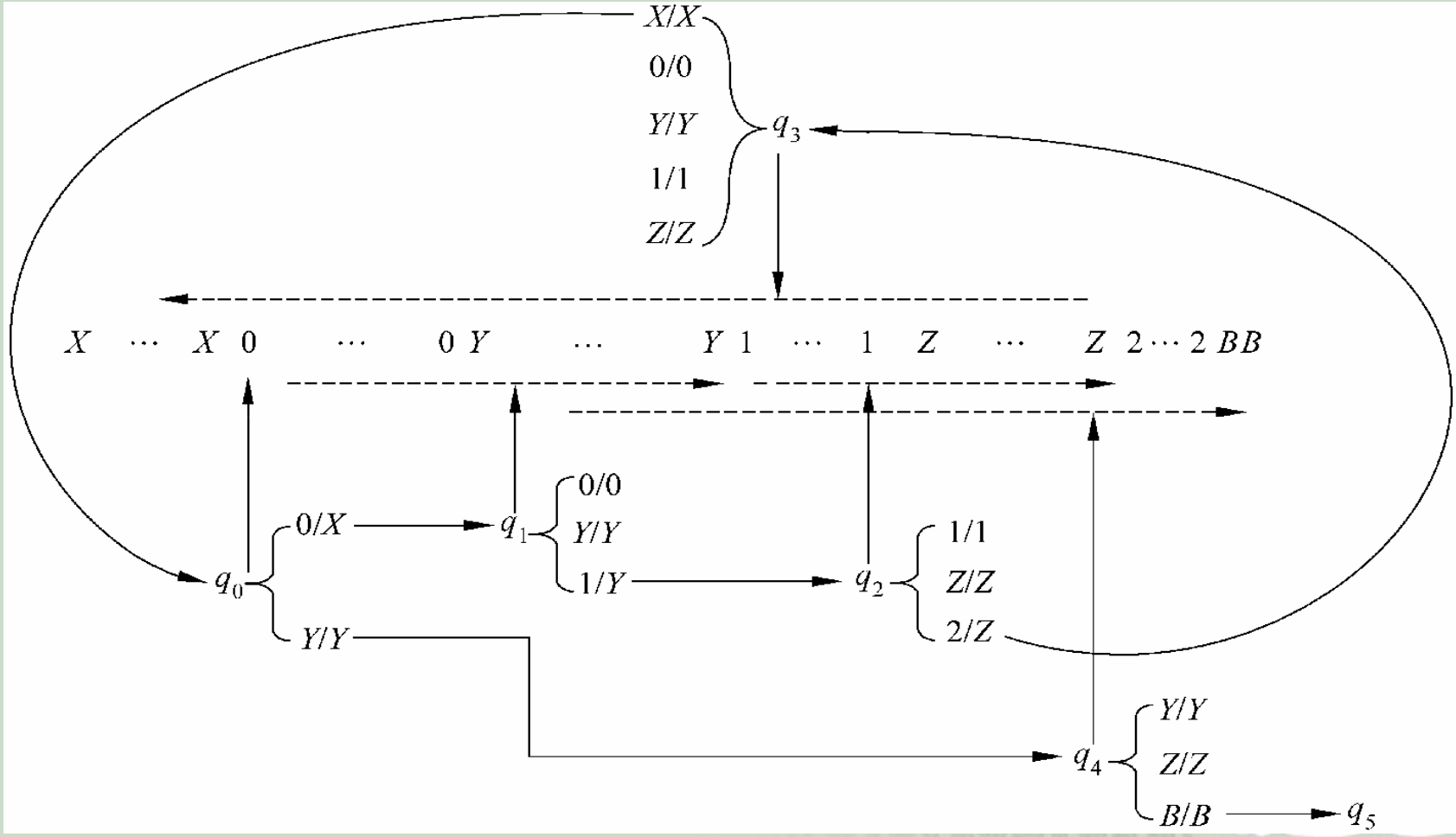
  X…X0…0Y…Y1…1Z…Z2…2BB

- 需要给予边界情况密切的关注。

基本TM

- 边界情况
- **X…XX…XY…YY…YZ…Z2…2BB**
- **X…XX…XY…Y1…1Z…Z2…2BB**
- **X…X0…0Y…YY…YZ…Z2…2BB**
- **X…X0…0Y…Y1…1Z…ZZ…ZBB**
- **X…X0…0Y…YY…YZ…ZZ…ZBB**
- **X…XX…XY…YY…YZ…ZZ…ZBB**

构造思路


X/X
0/0
Y/Y
1/1
Z/Z
q_3
X ⋯ X 0 ⋯ 0 Y ⋯ Y 1 ⋯ 1 Z ⋯ Z 2 ⋯ 2 BB
0/X
q_1
0/0
Y/Y
1/Y
q_0
q_2
1/1
Z/Z
2/Z
Y/Y
q_4
Y/Y
Z/Z
B/B
q_5

移动函数

	0	1	2	X	Y	Z	B
q_0	(q_1,X,R)				(q_4,Y,R)		
q_1	$(q_1,0,R)$	(q_2,Y,R)			(q_1,Y,R)		
q_2		$(q_2,1,R)$	(q_3,Z,L)			(q_2,Z,R)	
q_3	$(q_3,0,L)$	$(q_3,1,L)$		(q_0,X,R)	(q_3,Y,L)	(q_3,Z,L)	
q_4					(q_4,Y,R)	(q_4,Z,R)	(q_5,B,R)
q_5							

TM作为非负整函数的计算模型

- 非负整数进行编码 ——1进制
  - 用符号串0^n表示非负整数n。
- 用符号串

$$0^{n_1}10^{n_2}1\cdots 10^{n_k}$$

表示k元函数$f(n_1, n_2,\ldots, n_k)$的输入。

如果$f(n_1, n_2,\ldots, n_k)=m$，则该TM的输出为$0^m$。

TM作为非负整函数的计算模型

- **图灵可计算的(Turing computable)**
- 设有k元函数$f(n_1, n_2,\ldots, n_k)=m$，

  TM $M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$接受输入串

  $0^{n_1}10^{n_2}1\cdots10^{n_k}$

  输出符号串0^m；当$f(n_1, n_2,\ldots, n_k)$无定义时，TM M没有恰当的输出给出，则称TM M计算k元函数$f(n_1, n_2,\ldots, n_k)$，也称$f(n_1, n_2,\ldots, n_k)$为TM M计算的函数，也称f是图灵可计算的。

TM作为非负整函数的计算模型

- **完全递归函数(total recursive function)**

设有k元函数$f(n_1, n_2,\ldots, n_k)$，如果对于任意的$n_1, n_2,\ldots, n_k$，f均有定义，也就是计算f的TM总能给出确定的输出，则称f为完全递归函数。

- **部分递归函数(partial recursive function)**

TM计算的函数称为部分递归函数。

TM作为非负整函数的计算模型

- **例** 构造TM M_4，对于任意非负整数n、m，M_4计算m+n。

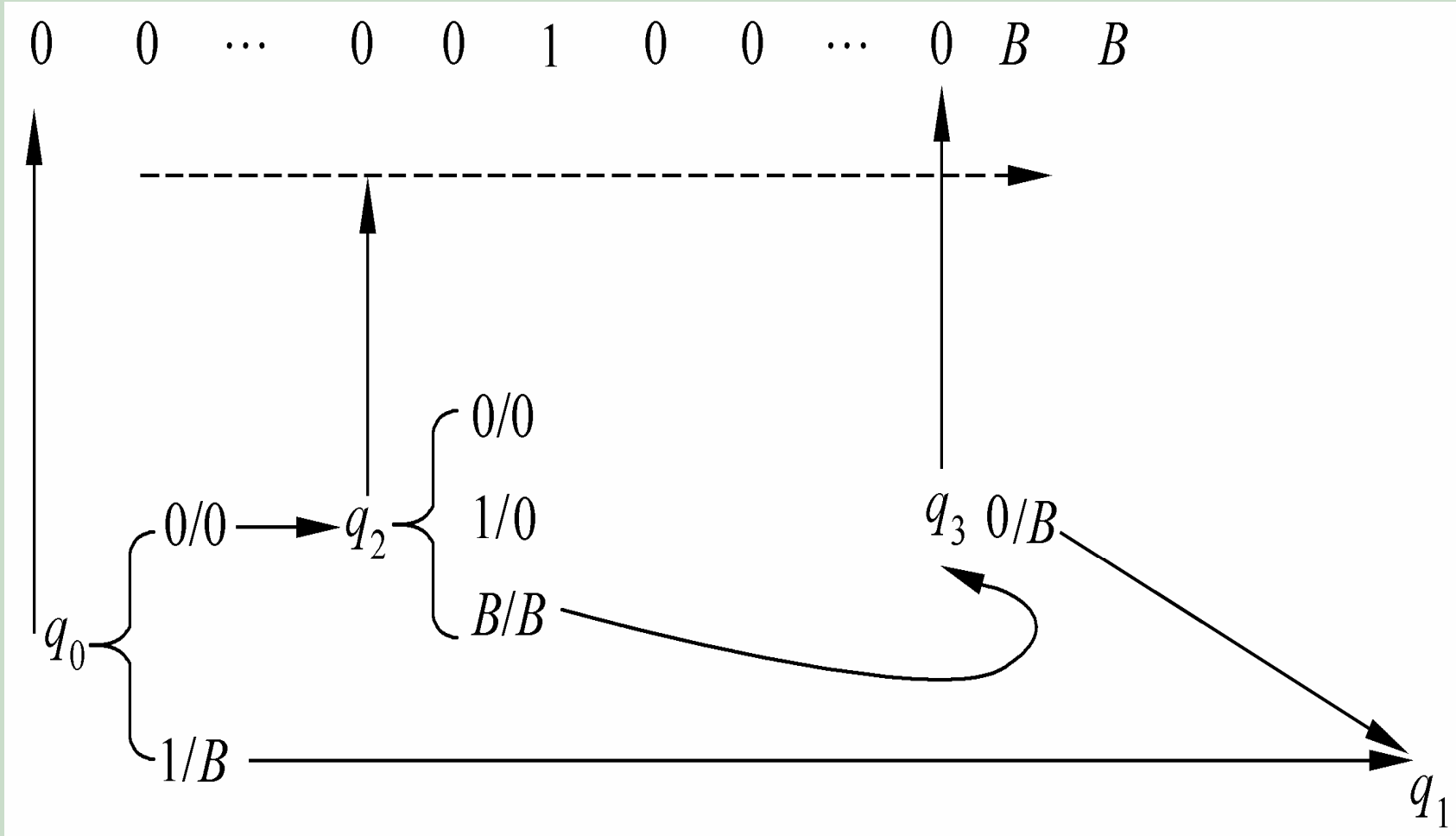
  分析：M_4的输入为0^n10^m，输出0^{n+m}的符号串。n和m为0的情况需要特殊考虑。

（1）当n为0时，只用将1变成B就完成了计算，此时无需考察m是否为0；

（2）当m为0时，需要扫描过表示n的符号0，并将1改为B。

（3）当n和m都不为0时，我们需要将符号1改为0，并将最后一个0改为B。

构造思路



M_4

$M_4=(\{q_0, q_1, q_2, q_3\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_1\})$

$\delta(q_0, 1)=(q_1, B, R)$

$\delta(q_0, 0)=(q_2, 0, R)$

$\delta(q_2, 0)=(q_2, 0, R)$

$\delta(q_2, 1)=(q_2, 0, R)$

$\delta(q_2, B)=(q_3, B, L)$

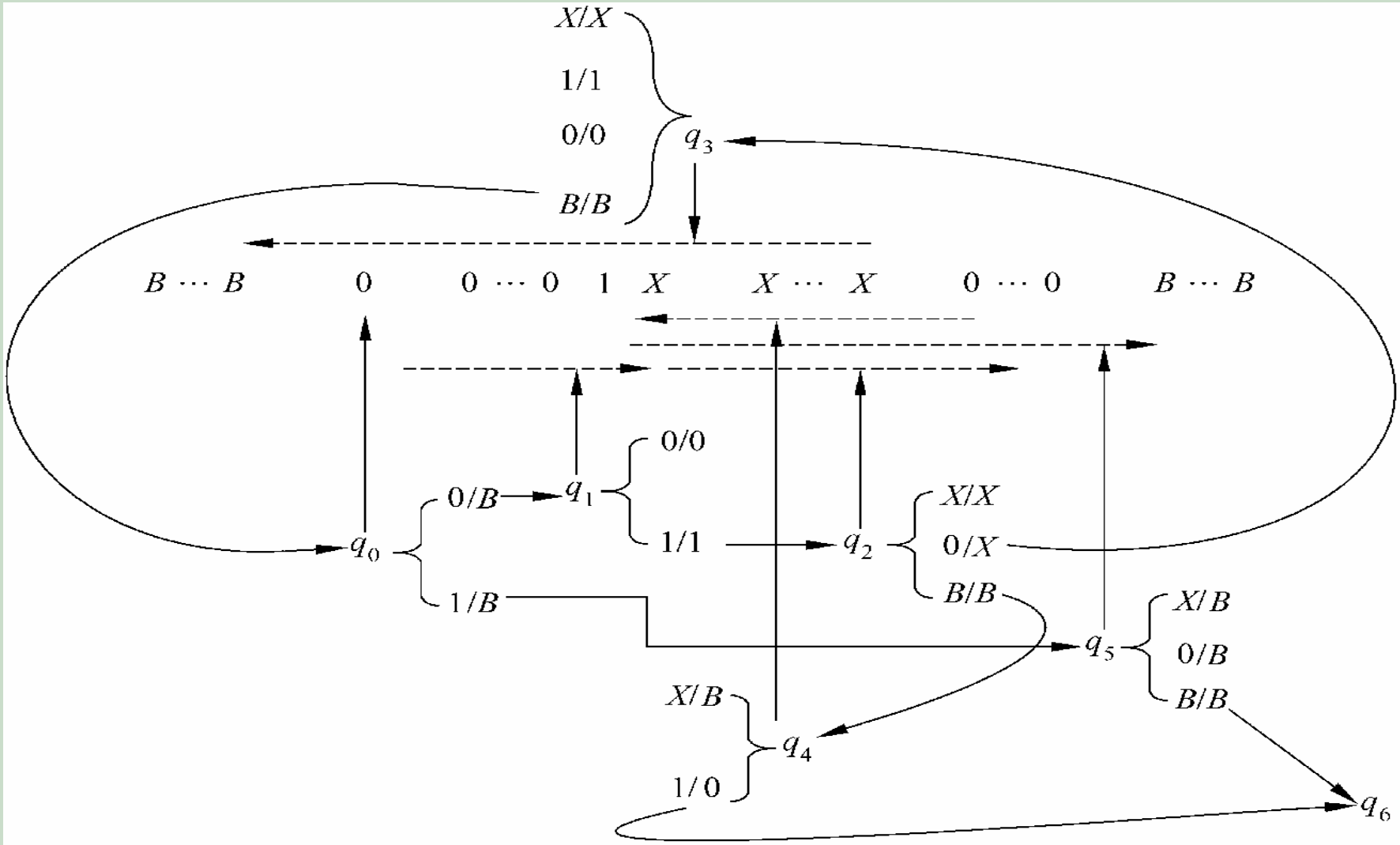
$\delta(q_3, 0)=(q_1, B, R)$

TM作为非负整函数的计算模型

- 例 构造TM M_5，对于任意非负整数n，m，M_5计算如下函数：

$$n \dot{-} m = \begin{cases} n-m & n \geqslant m \\ 0 & n < m \end{cases}$$

构造思路


X/X
1/1
0/0
B/B
q_3
B ··· B 0 0 ··· 0 1 X X ··· X 0 ··· 0 B ··· B
0/0
0/B
q_1
q_0
1/1
q_2
X/X
0/X
B/B
1/B
q_5
X/B
0/B
B/B
X/B
q_4
1/0
q_6

M_5

- $M_5=(\{q_0, q_1, q_2, q_3, q_4, q_5, q_6,\}, \{0,1\}, \{0, 1, X, B\}, \delta, q_0, B, \{q_6\})$

$\delta(q_0, 0)=(q_1, B, R)$

$\delta(q_0, 1)=(q_5, B, R)$

$\delta(q_1, 0)=(q_1, 0, R)$

$\delta(q_1, 1)=(q_2, 1, R)$

$\delta(q_2, X)=(q_2, X, R)$

$\delta(q_2, 0)=(q_3, X, L)$

$\delta(q_2, B)=(q_4, B, L)$

$\delta(q_3, X)=(q_3, X, L)$

$\delta(q_3, 1)=(q_3, 1, L)$

$\delta(q_3, 0)=(q_3, 0, L)$

$\delta(q_3, B)=(q_0, B, R)$

M_5

$\delta(q_4, X) = (q_4, B, L)$

$\delta(q_4, 1) = (q_6, 0, R)$

$\delta(q_5, X) = (q_5, B, R)$

$\delta(q_5, 0) = (q_5, B, R)$

$\delta(q_5, B) = (q_6, B, R)$。

TM的构造

1. 状态的有穷存储功能的利用

- 例 构造TM M_6，使得$L(M_6)=\{x \mid x \in \{0,1\}^* \& x$中至多含3个1$\}$。

  分析：M_6只用记录已经读到的1的个数。

  q[0]表示当前已经读到0个1；

  q[1]表示当前已经读到1个1；

  q[2]表示当前已经读到2个1；

  q[3]表示当前已经读到3个1。

状态的有穷存储功能的利用

- M_6=({q[0], q[1], q[2], q[3], q[f]}, {0,1}, {0,1,B}, δ , q[0], B, {q[f]})

δ (q[0], 0)=(q[0], 0, R)

δ (q[0], 1)=(q[1], 1, R)

δ (q[0], B)=(q[f], B, R)

δ (q[1], 0)=(q[1], 0, R)

δ (q[1], 1)=(q[2], 1, R)

δ (q[1], B)=(q[f], B, R)

δ (q[2], 0)=(q[2], 0, R)

δ (q[2], 1)=(q[3], 1, R)

δ (q[2], B)=(q[f], B, R)

δ (q[3], 0)=(q[3], 0, R)

δ (q[3], B)=(q[f], B, R)

状态的有穷存储功能的利用

- TM是要接受且仅接受恰含3个1的0、1串的TM，对M_6进行修改，得到M_7
- $L(M_7)$ ={x | x∈{0,1}* & x中含且仅含3个1}
- M_7=({q[0], q[1], q[2], q[3], q[f]}, {0, 1}, {0, 1, B}, δ , q[0], B, {q[f]})

$L(M_7)=\{x \mid x\in\{0,1\}^*$且 x中仅含3个1$\}$

δ (q[0], 0)=(q[0], 0, R)

δ (q[0], 1)=(q[1], 1, R)

δ (q[1], 0)=(q[1], 0, R)

δ (q[1], 1)=(q[2], 1, R)

δ (q[2], 0)=(q[2], 0, R)

δ (q[2], 1)=(q[3], 1, R)

δ (q[3], 0)=(q[3], 0, R)

δ (q[3], B)=(q[f], B, R)

$L(M_8)=\{x \mid x\in\{0,1\}^*\ \&\ x$中至少含3个1$\}$

M_8=({q[0], q[1], q[2], q[f]}, {0, 1}, {0, 1, B}, δ, q[0], B, {q[f]})

δ (q[0], 0)=(q[0], 0, R)

δ (q[0], 1)=(q[1], 1, R)

δ (q[1], 0)=(q[1], 0, R)

δ (q[1], 1)=(q[2], 1, R)

δ (q[2], 0)=(q[2], 0, R)

δ (q[2], 1)=(q[f], 1, R)

状态的有穷存储功能的利用

- **例** 构造TM M_9它的输入字母表为{0，1}，现在要求M_9在它的输入符号串的尾部添加子串101。

  分析：

  - 将待添加子串101存入有穷控制器。
  - 首先找到符号串的尾部。
  - 将给定符号串中的符号依次地印刷在输入带上
  - 每印刷一个符号，就将它从有穷控制器的“存储器”中删去，当该“存储器”空时，TM就完成了工作。

状态的有穷存储功能的利用

M_9=({q[101], q[01], q[1], q[ε]}, {0, 1}, {0, 1, B}, δ , q[101], B, {q[ε]})

其中 δ 的定义为：

δ (q[101], 0)=(q[101], 0, R)

δ (q[101], 1)=(q[101], 1, R)

δ (q[101], B)=(q[01], 1, R)

δ (q[01], B)=(q[1], 0, R)

δ (q[1], B)=(q[ε], 1, R)

状态的有穷存储功能的利用

- **例** 构造TM M_{10}它的输入字母表为{0，1}，要求M_{10}在它的输入符号串的开始处添加子串101。
- 将有穷控制器中的“存储器”分成两部分
  - 第一部分用来存放待添加的子串。
  - 第二部分用来存储因添加符号串当前需要移动的输入带上暂时无带方格存放的子串。
  - 一般形式为q[x, y]
    - x待添加子串
    - y当前需要移动的输入带上暂时无带方格存放的子串。

状态的有穷存储功能的利用

- **q[x, ε]为开始状态；**
- **q[ε , ε]为终止状态。**
- **设a、b为输入符号**
- **δ (q[ax, y], b) = (q[x, yb], a, R)**
  - 表示在没有完成待插入子串的印刷之前，要将待插入子串的首字符印刷在**TM**当前扫描的带方格上。

状态的有穷存储功能的利用

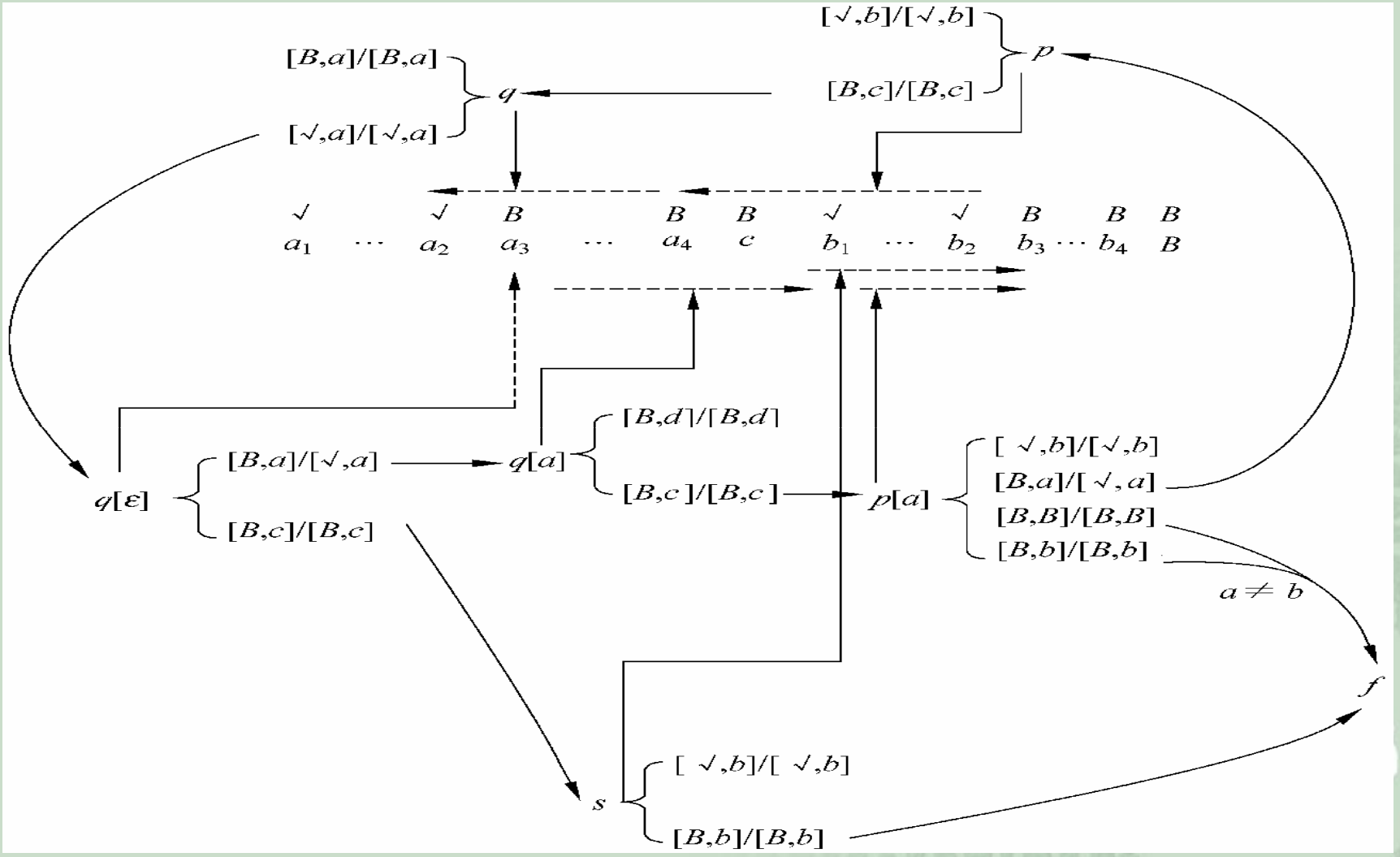
- **δ (q[ε ,a y], b) = (q[ε , yb], a, R)**
  - 表示当完成待插入子串的插入工作之后，必须将插入点之后的子串顺序地向后移动。
- **δ (q[ε ,a y], B) = (q[ε , y], a, R)**
  - 表示读头当前所指的带方格为空白，现将“存储器”的第二部分中的当前首符号a印刷在此带方格上，同时将这个符号从存储器中删除。

多道(multi-track)技术

- 例 构造M_{11}，使$L(M_{11})=\{xcy \mid x,y\in\{0,1\}^+ 且\ x\neq y\}$。

  分析：
  - 以符号c为分界线，逐个地将c前的符号与c后的符号进行比较。
  - 当发现对应符号不同时，就进入终止状态。
  - 当发现x与y的长度不相同而进入终止状态。
  - 发现它们相同而停机。
  - 一个道存放被检查的符号串，另一个存放标记符。

构造思路


[√,b]/[√,b]
[B,c]/[B,c]
p
[B,a]/[B,a]
[√,a]/[√,a]
q
√ √ B B B √ √ B B B
a1 ⋯ a2 a3 ⋯ a4 c b1 ⋯ b2 b3 ⋯ b4 B
[B,d]/[B,d]
[B,a]/[√,a]
q[a]
[B,c]/[B,c]
q[ε]
[B,c]/[B,c]
p[a]
[√,b]/[√,b]
[B,a]/[√,a]
[B,B]/[B,B]
[B,b]/[B,b]
a ≠ b
f
[√,b]/[√,b]
s
[B,b]/[B,b]

多道(multi-track)技术

- M_{11}=({q[ε], q[0], q[1], p[0], p[1] , q, p, s, f}, {[B,0], [B,1], [B,c]}, {[B,0], [B,1], [B,c], [√,0], [√,1], [B,B]}, δ , q[ε], [B,B], {f})

δ (q[ε], [B,a])=(q[a], [√,a], R)

δ (q[a], [B,d])=(q[a], [B,d], R)

δ (q[a], [B,c])=(p[a], [B,c], R)

δ (p[a], [√,b])=(p[a], [√,b], R)

δ (p[a], [B,a])=(p, [√,a], L)

多道(multi-track)技术

δ (p, [√,b])=(p, [√,b], L)
δ (p, [B,c])=(q, [B,c], L)
δ (q, [B,a])=(q, [B,a], L)
δ (q, [√,a])=(q[ε], [√,a], R)
δ (p[a], [B,b])=(f, [B,b],R)
δ (p[a], [B,B])=(f, [B,B], R)
δ (q[ε], [B,c])=(s, [B,c], R)
δ (s, [√,b])=(s, [√,b], R)
δ (s, [B,a])=(f, [B,a], R)

子程序(subroutine)技术

- 将TM的设计看成是一种特殊的程序设计，将子程序的概念引进来。
- 一个完成某一个给定功能的TM M′从一个状态q开始，到达某一个固定的状态f结束。
- 将这两个状态作为另一个TM M的两个一般的状态。
- 当M进入状态q时，相当于启动M′(调用M′对应的子程序)；当M′进入状态f时，相当于返回到M的状态f。

子程序(subroutine)技术

- 例 构造M_{12}完成正整数的乘法运算。

  分析:

  - 设两个正整数分别为m和n。
  - 输入串为0^n10^m 。
  - 输出应该为0^{n*m} 。
  - 算法思想：每次将n个0中的1个0改成B，就在输入串的后面复写m个0。
  - 在M_{12}的运行过程中，输入带的内容为

    $$B^h0^{n-h}10^m10^{m*h}B$$

正整数的乘法运算

（1）初始化。完成将第一个0变成B，并在最后一个0后写上1。我们用q_0表示启动状态，用q_1表示完成初始化后的状态。首先，消除前n个0中的第一个0，

$$q_0 0^n 1 0^m \vdash^+ B q_1 0^{n-1} 1 0^m 1$$

（2）主控系统。从状态q_1开始，扫描过前n个0中剩余的0和第一个1，将读头指向m个0的第一个，此时的状态为q_2。其ID变化为

$$B^h q_1 0^{n-h} 1 0^m 1 0^{m*(h-1)} B \vdash^+ B^h 0^{n-h} 1\, q_2 0^m 1 0^{m*(h-1)} B$$

正整数的乘法运算

- 当子程序完成m个0的复写后，回到q_3。这个状态相当于子程序的返回（终止）状态。然后在q_3状态下，将读头移回到前n个0中剩余的0中的第一个0，并将这个0改成B，进入q_1状态，准备进行下一次循环

$$B^h0^{n-h}1\ q_30^m10^{m*h}B \vdash^+ B^{h+1}q_10^{n-h-1}10^m10^{m*h}B$$

正整数的乘法运算

- 当完成m*n个0的复写之后，清除输入带上除了这m*n个0以外的其他非空白符号。q_4为终止状态

$$B^n q_1 10^m 10^{m*n} B \vdash^+ B^{n+1+m+1} q_4 0^{m*n} B$$

（3）子程序。完成将m个0复写到后面的任务。从q_2启动，到q_3结束，返回到主控程序。

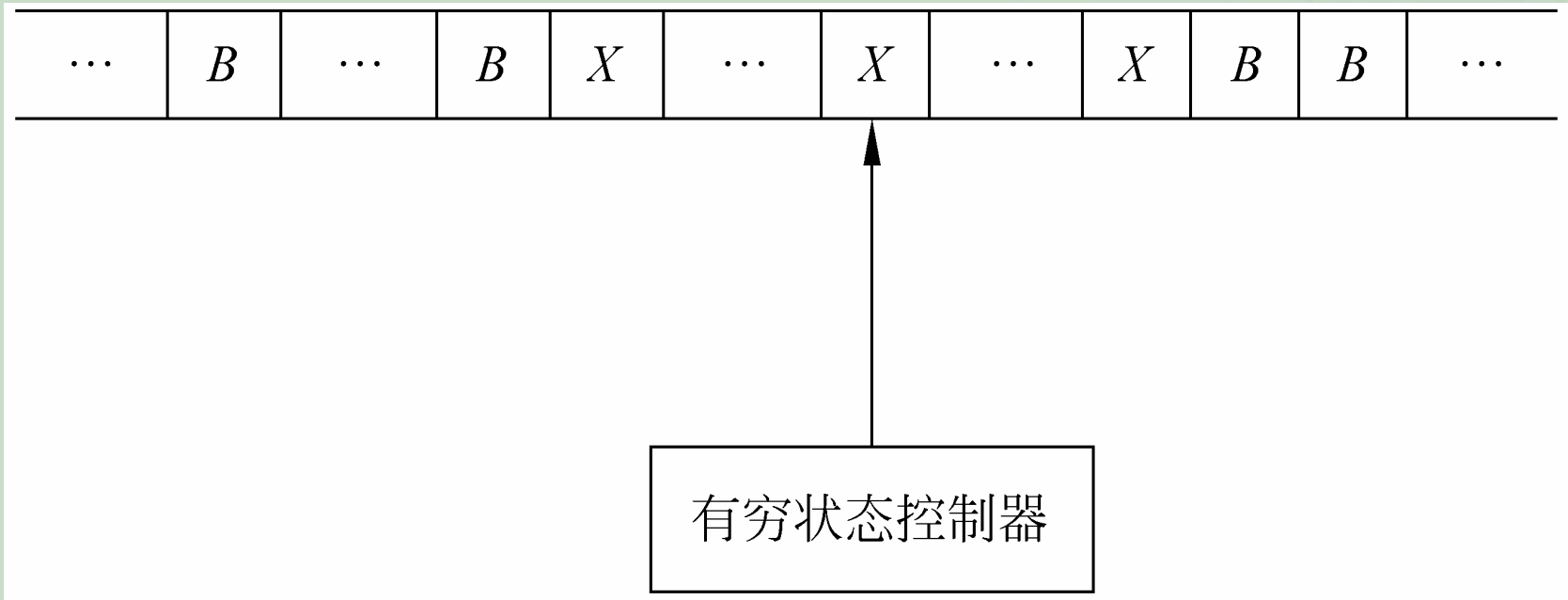
$$B^{h+1} 0^{n-h-1} 1\ q_2 0^m 10^{m*h} B \vdash^+ B^{h+1} 0^{n-h-1} 1\ q_3 0^m 10^{m*h+1} B$$

TM的变形

- 从不同的方面对TM进行扩充。
  - 双向无穷带TM。
  - 多带TM。
  - 不确定的TM。
  - 多维TM等。
- 它们与基本的TM等价。

双向无穷带TM

物理模型



双向无穷带TM

- 双向无穷带 (Turing machine with two-way infinite tape，TM)

$$\text{TM } M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

- Q, Σ, Γ, δ ,q_0 , B , F的意义同基本图灵机定义。
- M的即时描述ID同基本图灵机定义。
- 允许M的读头处在输入串的最左端时，仍然可以向左移动。

双向无穷带TM

- M的当前ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$
- 如果 $\delta(q, X_i)=(p, Y, R)$
  - 当$i\neq 1$或者$Y\neq B$时，M的下一个ID为

    $X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$
  - 记作

$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash_M X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$

  - 表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$下，经过一次移动，将ID变成$X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$。

双向无穷带TM

- 当i=1并且Y=B时，M的下一个ID为

  $pX_2 \ldots X_n$

- 记作

  $qX_1X_2 \ldots X_n \vdash_M pX_2 \ldots X_n$

- 这就是说，和基本TM在读头右边全部是B时，这些B不在ID中出现一样，当双向无穷带TM的读头左边全部是B时，这些B也不在该TM的ID中出现。

双向无穷带TM

- 如果 $\delta(q, X_i)=(p, Y, L)$
  - 当$i \neq 1$时，M的下一个ID为

    $X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$
  - 记作

$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash_M X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$

  - 表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots_n$下，经过一次移动，将ID变成$X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$。

双向无穷带TM

- 当i=1时，M的下一个ID为

  $pBYX_2\ldots X_n$

- 记作

  $qX_1X_2\ldots X_n \vdash_M pBYX_2\ldots X_n$

- 表示M在ID $qX_1X_2\ldots X_n$下，经过一次移动，将ID变成$pBYX_2\ldots X_n$。

双向无穷带TM

定理 对于任意一个双向无穷带TM M，存在一个等价的基本TM M′。

证明要点：

- 双向无穷存储的模拟：用一个具有2个道的基本TM来模拟：一个道存放M开始启动时读头所注视的带方格及其右边的所有带方格中存放的内容；另一个道按照相反的顺序存放开始启动时读头所注视的带方格左边的所有带方格中存放的内容。
- 双向移动的模拟：在第1道上，移动的方向与原来的移动方向一致，在第2道上，移动的方向与原来的移动方向相反。

用单向无穷带模拟双向无穷带

…	B	A_{-n}	…	A_{-1}	A_0	…	A_i	…	A_m	B	B	…

(a) M的双向无穷带

A_0	A_1	…	A_i	…	B	…
¢	A_{-1}	…	A_{-i}	…	B	…

(b) M'用单向无穷带模拟 M的双向无穷带

多带TM

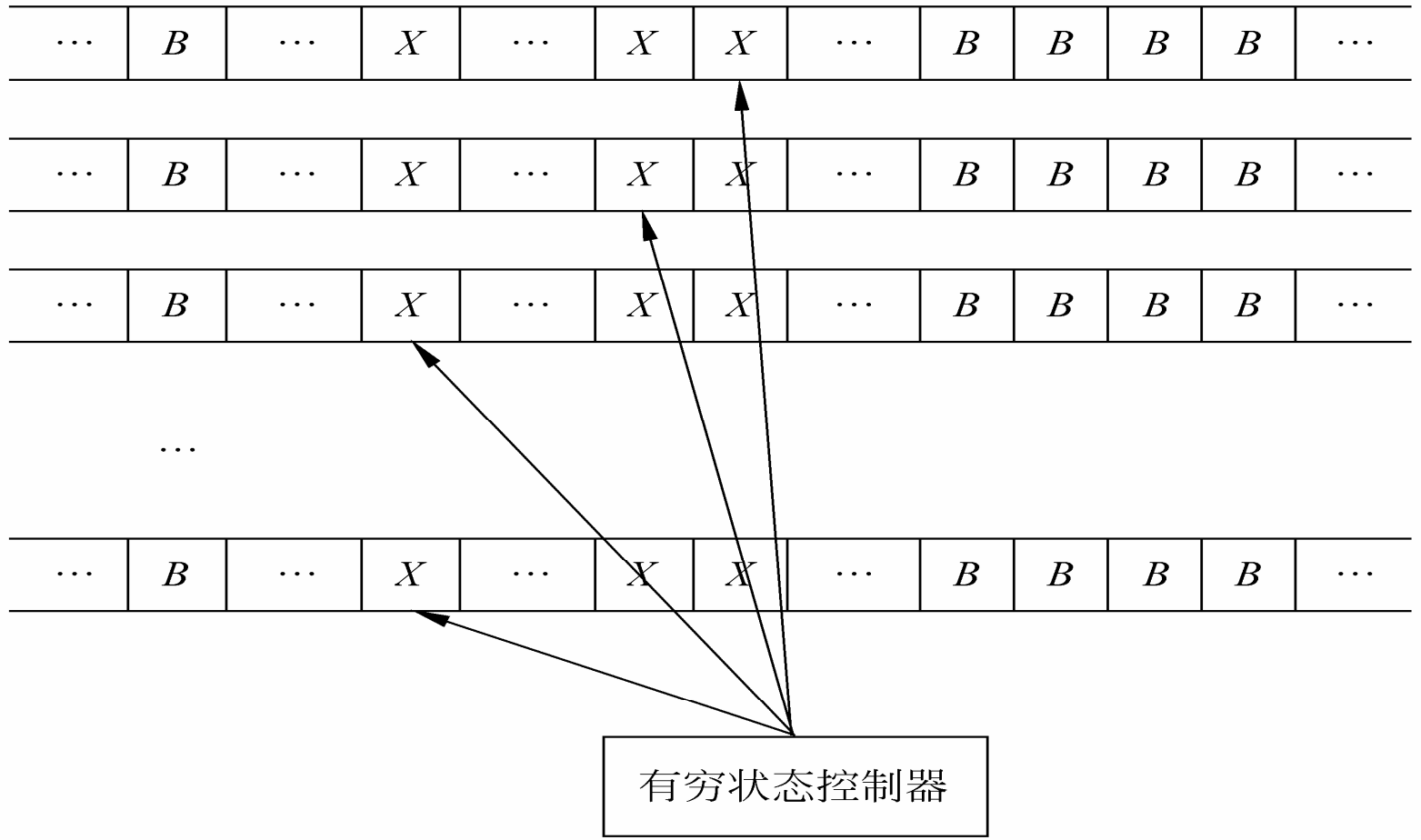
- **多带TM(multi-tape turing machine)**
- 允许**TM**有多个双向无穷带，每个带上有一个相互独立的读头。
- **k**带**TM**在一次移动中完成如下三个动作

(1) 改变当前状态；

(2) 各个读头在自己所注视的带方格上印刷一个希望的符号。

(3) 各个读头向各自希望的方向移动一个带方格。

多带TM



多带TM

定理 多带TM与基本的TM等价。

证明要点：

- 对一个k带TM，用一条具有2k道的双向无穷带TMM′，实现对这个k带TMM的模拟。
- 对应M的每一条带，M′用两个道来实现模拟。其中一条道用来存放对应的带的内容，另一条道专门用来标记对应带上的读头所在的位置。

不确定的TM

- 不确定TM与基本TM的区别是对于任意的$(q，X)\in Q\times\Gamma$，

$\delta(q，X)=\{(q_1，Y_1，D_1)，(q_2，Y_2，D_2)，\ldots，(q_k，Y_k，D_k)\}$

- D_j为读头的移动方向。即$D_j\in\{R，L\}$。
- 表示M在状态q，读到X时，可以有选择地进入状态q_j，印刷字符Y_j，按D_j移动读头
- $L(M)=\{w\mid w\in\Sigma^*$且$ID_1\vdash^*ID_n$，且$ID_n$含M的终止状态}。

不确定的TM

定理 不确定的TM与基本的TM等价

- 证明要点：
  - 让等价的基本TMM′ 具有3条带。
  - 第1条带用来存放输入。
  - 第2条带上系统地生成M的各种可能的移动序列
  - M′在第3条带上按照第2条带上给出的移动系列处理输入串，如果成功，则接受之，如果不成功，则在第2条带上生成下一个可能的移动系列，开始新一轮的“试处理”。

多维TM

- **多维TM(multi-dimensional Turing machine)**
  - 读头可以沿着多个维移动。
- **k维TM(k-dimensional Turing machine)**
- **TM可以沿着k维移动。**
- **k维TM的带由k维阵列组成，而且在所有的2k个方向上都是无穷的，它的读头可以向着2k个方向中的任一个移动。**

多维TM

定理　多维TM与基本TM等价。

- 用一维的形式表示k维的内容，就像多维数组在计算机的内存中都被按照一维的形式实现存储一样。
- **段(Segment)**用来表是一维上的内容。
- 用#作为段分割符。
- ¢用作该字符串的开始标志，$用作该字符串的结束标志。

基本TM模拟2维TM

B	a_1	a_2	a_3	a_4	B	B	B	B	B	B
B	a_5	B	a_6	a_7	a_8	a_9	a_{10}	B	B	B
B	a_{11}	B	B	B	B	a_{12}	B	a_{13}	B	a_{14}
a_{15}	a_{16}	B	B	B	B	B	B	B	B	a_{16}
B	B	B	a_{17}	B	B	B	B	B	a_{18}	B
a_{19}	a_{20}	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B	a_{21}

基本TM模拟2维TM

$\not\subset Ba_1a_2a_3a_4BBBBBB \# Ba_5Ba_6a_7a_8a_9a_{10}BBB \#$

$Ba_{11}BBBBa_{12}Ba_{13}Ba_{14} \# a_{15}\ a_{16}BBBBBBBB\ a_{16}\#$

$BBB\ a_{17}BBBBBa_{18}B \# a_{19}a_{20}BBBBBBBBB \#$

$BBBBBBBBBBa_{21}\$$

其他TM

1. 多头TM
2. 离线TM
3. 作为枚举器的TM
4. 多栈机
5. 计数机
6. Church-Turing论题与随机存取机

多头TM

- **多头TM(multi-head Turing machine)**
- 指在一条带上有多个读头，它们受M的有穷控制器的统一控制，M根据当前的状态和这多个头当前读到的字符确定要执行的移动。在M的每个动作中，各个读头所印刷的字符和所移动的方向都可以是相互独立的m

多头TM

定理 多头TM与基本的TM等价。

- 可以用一条具有k+1个道的基本TM来模拟一个具有k个头的TM（k头TM）。其中一个道用来存放原输入带上的内容，其余k个道分别用来作为k个读头位置的标示。

离线TM

- **离线TM(off-line Turing machine)**
  - 有一条输入带是只读带**(read-only tape)**的多带**TM**。
- 符号¢和**$**用来限定它的输入串存放区域，¢在左边，**$**在右边。
- 不允许该带上的读头移出由¢和**$**限定的区域——离线的**TM**。
- 如果只允许只读带上的读头从左向右移动，则称之为**在线TM(on-line Turing machine)**。

离线TM

定理 离线TM与基本的TM等价。

证明要点：让模拟M的离线TM比M多一条带，并且用这多出来的带复制M的输入串。然后将这条带看作是M的输入带，模拟M进行相应的处理。

作为枚举器的TM

- **作为枚举器的TM(Turing machine as enumerator)**
  - 多带TM，其中有一条带专门作为输出带，用来记录产生语言的每一个句子。
- 在枚举器中，一旦一个字符被写在了输出带上，它就不能被更改。如果该带上的读头的正常移动方向是向右移动的话，这个带上的读头是不允许向左移动的。
- 如果这个语言有无穷多个句子，则它将永不停机。它每产生一个句子，就在其后打印一个分割符“#”。
- 枚举器产生的语言记为G(M)。

作为枚举器的TM

- **规范的顺序(canonical order)**

定理 L为递归可枚举语言的充分必要条件是存在一个TM M，使得L=G(M)。

定理 一个语言L为递归语言的充分必要条件是存在一个TMM，使得L=G(M)，并且L是被M按照规范顺序产生的。

多栈机

- **多栈机(multi-stack machines)**是一个拥有一条只读输入带和多条存储带的不确定**TM**。
  - 多栈机的只读带上的读头不能左移。
  - 存储带上的读头可以向左和向右移动。
    - 右移时，一般都在当前注视的带方格上印刷一个非空白字符
    - 左移时，必须在当前注视的带方格中印刷空白字符**B**。
- 一个确定的**双栈机(double stack machines)**是一个确定的**TM**，它具有一条只读的输入带和两条存储带。存储带上的读头左移时，只能印刷空白符号**B** 。

多栈机

- 下推自动机是一种非确定的多带TM。它有一条只读的输入带，一条存储带。

定理 一个任意的单带TM可以被一个确定的双栈机模拟。

计数机

- **计数机(counter machine)**
  - 有一条只读输入带和若干个用于计数的单向无穷带的离线TM。
  - 拥有n个用于计数带的计数机被称为n计数机。
  - 用于计数的带上仅有两种字符，一个为相当于是作为栈底符号的Z，该字符也可以看作是计数带的首符号，它仅出现在计数带的最左端；另一个就是空白符B。这个带上所记的数就是从Z开始到读头当前位置所含的B的个数。

定理 TM可以被一个双计数机模拟。

丘奇-图灵论题与随机存取机

- **丘奇-图灵论题** 对于任何可以用有效算法解决的问题，都存在解决此问题的**TM**。
- **随机存取机(random access machine，RAM)**含有无穷多个存储单元，这些存储单元被按照**0、1、2、…**进行编号，每个存储单元可以存放一个任意的整数；有穷个能够保存任意整数的算术寄存器。这些整数可以被译码成通常的各类计算机指令。

定理 如果**RAM**的基本指令都能用**TM**来实现，那么就可以用**TM**实现**RAM**。

通用TM

- 通用TM(universal Turing machine)
  - 实现对所有TM的模拟。
- 编码系统
  - 它可以在实现对TM的表示的同时，实现对该TM处理的句子的表示。
  - 用0和1对这些除空白符以外的其他的带符号进行编码。同时也可以用0和1对TM的移动函数进行编码。

通用TM

- $M=(\{q_1, q_2, \ldots, q_n\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$
- 用X_1、X_2、X_3分别表示0、1、B，用D_1、D_2分别表示R、L。
- $\delta(q_i, X_j)=(q_k, X_l, D_m)$可以用$0^i10^j10^k10^l10^m$表示。

通用TM

- **M可用**

  111 $code_1$ 11 $code_2$ 11 …… 11 $code_r$ 111
- **$code_t$ 是动作 $\delta(q_i, X_j)=(q_k, X_l, D_m)$的形如 $0^i10^j10^k10^l10^m$的编码。**
- **TM M和它的输入串w则可以表示成**

  111 $code_1$ 11 $code_2$ 11 …… 11 $code_r$ 111w
- 按照规范顺序分别对表示TM的符号行和表示输入的符号行进行排序。

通用TM

- **L_d={w | w是第j个句子，并且第j个TM不接受它}** 不是递归可枚举语言。
- **通用语言(universal language)**
  - **L_u={<M，w> | M接受w}**
  - **<M，w>** 为如下形式的串，表示**TMM=({q_1, q_2, …, q_n}, {0，1}, {0，1，B}, δ ,q_1 , B , {q_2})**和它的输入串**w**。

  111 $code_1$ 11 $code_2$ 11 … 11 $code_r$ 111w
- 通用**TM**就是接受通用语言的图灵机。

通用TM

- 例 设TM $M_2=(\{ q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, B\}, \delta , q_4 , B ,\{q_3\})$，其中δ的定义如下:

$\delta (q_4, 0)= (q_4, 0, R)$

$\delta (q_4, 1)= (q_1, 1, R)$

$\delta (q_1, 0)= (q_1, 0, R)$

$\delta (q_1, 1)= (q_2, 1, R)$

$\delta (q_2, 0)= (q_2, 0, R)$

$\delta (q_2, 1)= (q_3, 1, R)$

通用TM

- 编码为

  1110000101000010101100001001010010110101010110100100100101100101001010011001000100010010111

- 通用TM检查M是否接受字符串**001101110**

  1110000101000010101100001001010010110101010110100100100101100101001010011001000100010010111001101110

几个相关的概念

- **可计算性(computability)**理论是研究计算的一般性质的数学理论。计算的过程就是执行算法的过程。
- 可计算理论的中心问题是建立计算的数学模型，研究哪些是可计算的，哪些是不可计算的。
- 可计算理论又称为**算法理论(algorithm theory)**。
- 在直观意义下，算法具有有限性、机械可执行性、确定性、终止性等特征。
- 可计算问题可以等同于图灵可计算问题。

几个相关的概念

- **可判定的(decidable)问题**
  - 它对应的语言是递归的。
- **不可判定的(undecidable)**
  - 没有这样的算法，它以问题的实例为输入，并能给出相应的“是”与“否”的判定。
- 递归语言举例

（1）L_{DFA}={<M, w> | M是一个DFA，w是字符串，M接受w}。

（2）L_{NFA}={<M, w> | M是一个NFA，w是字符串，M接受w}。

几个相关的概念

（3）L_{RE}={<r, w> | r是一个RE，w是字符串，w是r的一个句子}。

（4）E_{DFA}={<M> | M是一个DFA，且L(M)= Φ}。

（5）EQ_{DFA}={<M_1, M_2> | M_1, M_2是DFA，且L(M_1)=L(M_2)}。

（6）L_{CFG}={<G, w> | G是一个CFG，w是字符串，G产生w}。

（7）E_{CFG}={<G> | G是一个CFG，且L(G)= Φ}。

几个相关的概念

- **P类问题(class of P)**
  - **P**表示确定的**TM**在多项式时间(步数)内可判定的语言类。这些语言对应的问题称为是**P类问题**，这种语言称为多项式可判定的。
  - 例如，判定一个字符串是否为一个上下文无关语言的句子的问题就是**P**类问题。

几个相关的概念

- **NP类问题(class of NP)**
- **NP**表示不确定的**TM**在多项式时间(步数)内可判定的语言类。这些语言对应的问题称为是**NP**类问题，也称这些问题是**NP**复杂的，或者**NP**困难的。
- 这种语言称为非确定性多项式可判定的。
- **P=NP**？是理论计算机科学和当代数学中最大的悬而未决的问题之一。

几个相关的概念

- **NP完全的问题(NP complete problem)**
  - NP类中有某些问题的复杂性与整个类的复杂性相关联。如果能找到这些问题中的任何一个的多项式时间判定算法，那么，所有的NP问题都是多项式时间可以判定的。
  - TSP(旅行商问题)。
  - 可满足性问题。

小结

TM是一个计算模型，用**TM**可以完成的计算被称为是图灵可计算的。

(1) TM的基本概念：形式定义、递归可枚举语言、递归语言、完全递归函数、部分递归函数。

(2) 构造技术：状态的有穷存储功能的利用、多道技术、子程序技术。

小结

(3) TM的变形：双向无穷带TM、多带TM、不确定的TM、多维TM、多头TM、离线TM，它们都与基本TM等价。

(4) Church-Turing论题：对于任何可以用有效算法解决的问题，都存在解决此问题的TM。

(5) 通用TM可以实现对所有TM的模拟。

(6) 可计算性、P-NP问题。

习题

1 设有M=($\{q_0, q_1, q_2, q_3\}$, $\{0, 1\}$, $\{0, 1, B\}$, δ, q_0, B, $\{q_3\}$)，其中δ的定义如下：

$\delta(q_0, 0) = (q_0, 0, R)$

$\delta(q_0, 1) = (q_1, 1, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, 1) = (q_2, 1, R)$

$\delta(q_2, 0) = (q_2, 0, R)$

$\delta(q_2, 1) = (q_3, 1, R)$

请给出M处理输入串000101101的过程中经历的ID变换序列。

习题

2 设计识别下列语言的图灵机。

(1) $\{01^n0^{2m}1^n \mid m, n \geqslant 1\}$。

(2) 含有相同个数0和1的所有0、1串。

(3) $\{ww^T \mid w \in \{0, 1\}^*\}$。

3 设计计算下列函数的图灵机。

(1) n^2。

(2) $f(w)=ww^T$, 其中$w \in \{0, 1\}^+$。