第十二讲

- 图灵机与递归可枚举语言

图灵机与递归可枚举语言

- 图灵机的概念与定义
- 递归可枚举语言
- 递归语言
- 基本图灵机的几种编程技巧
- 对基本图灵机的扩展
- 受限的图灵机
- 图灵机与计算机

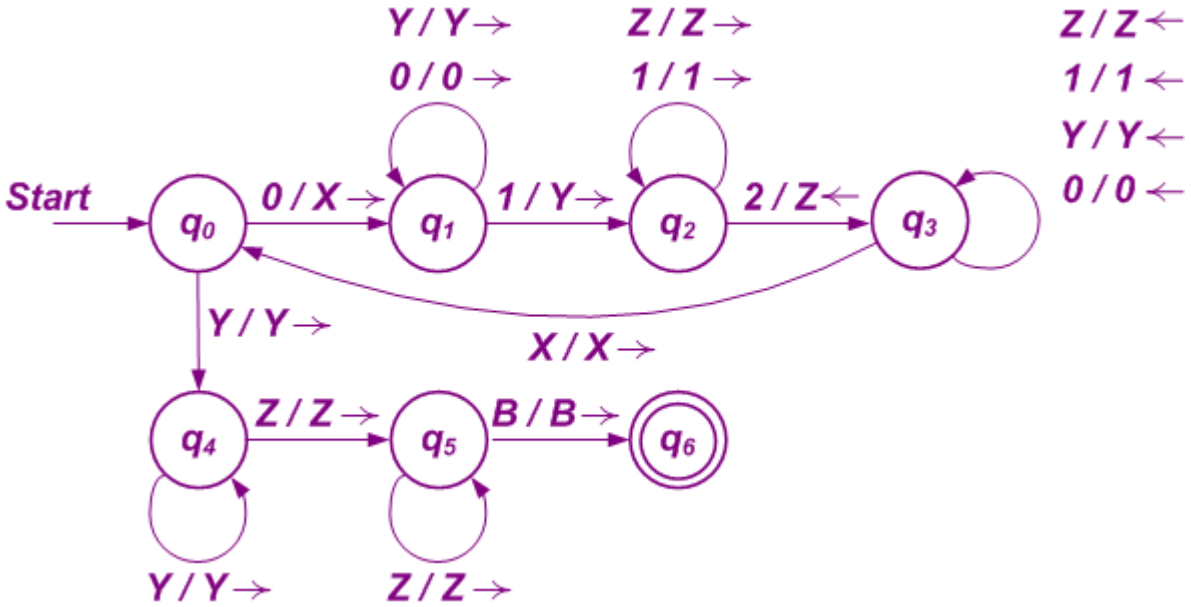
图灵机的概念与定义

✧ 回顾

设 $\Sigma = \{ 0, 1, 2 \}$， $L = \{ 0^n1^n2^n \mid n \geq 1 \}$.

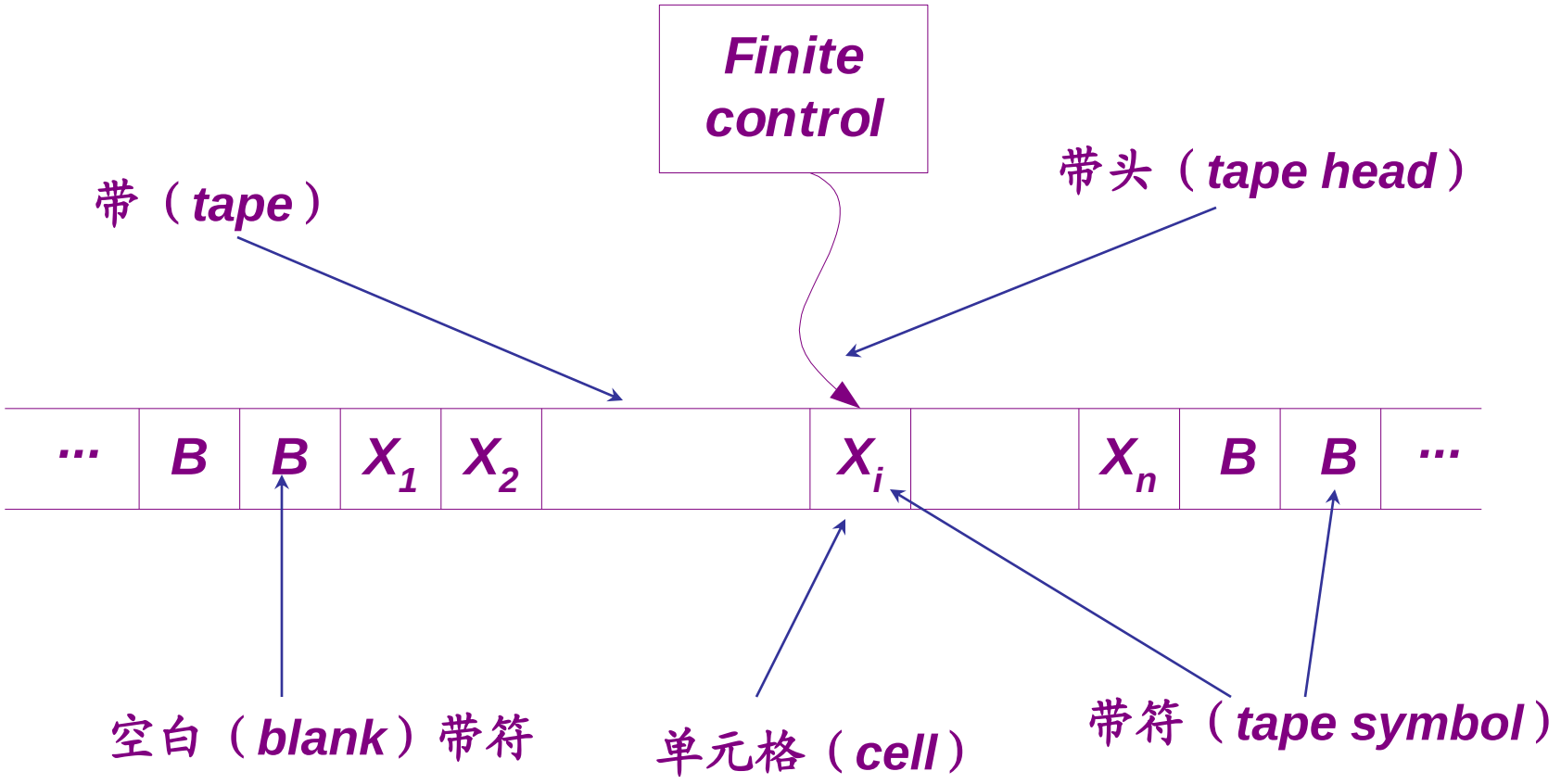
无有限自动机和上下文无关文法能够接受语言L.

存在一个图灵机可接受该语言.



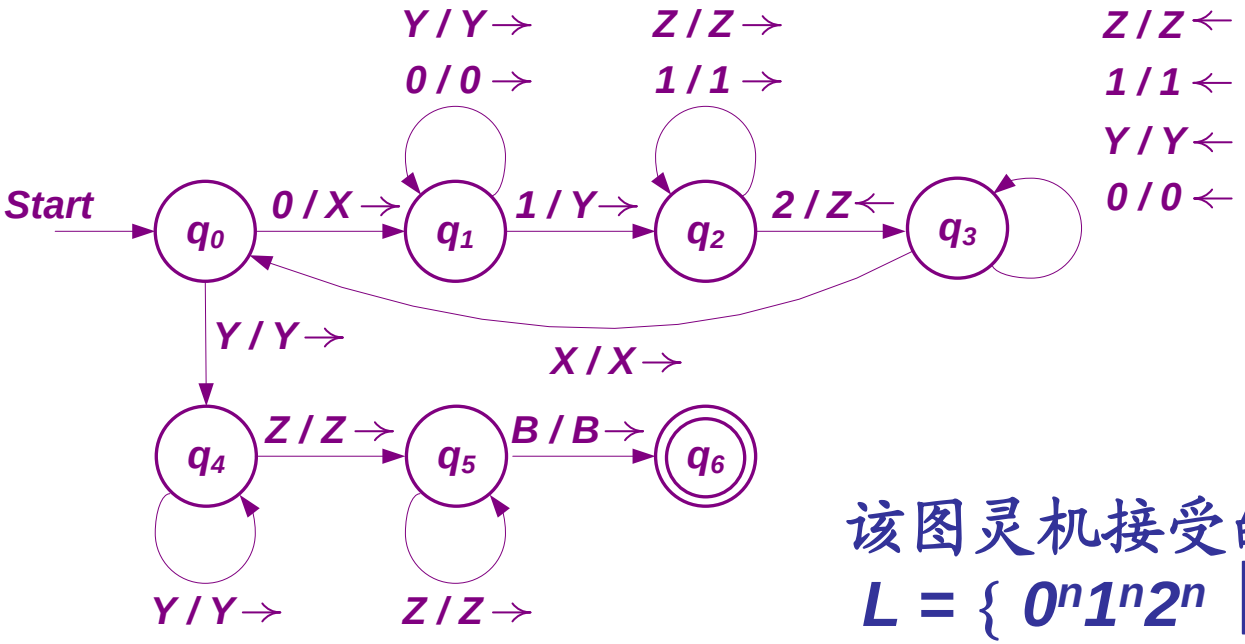
图灵机的概念与定义

✧ 基本图灵机



图灵机的概念与定义

✧ 举例



该图灵机接受的语言为

$L = \{ 0^n 1^n 2^n \mid n \geq 1 \}$.

…	B	B	0	0	1	1	2	2	B	B	…
			X	X	Y	Y	Z	Z			
	q_3		q_0	q_1	q_1	q_2	q_2	q_3	q_5	q_6	

图灵机的概念与定义

✧ 形式定义 一个图灵机 *TM (Turing machine)* 是一个七元组

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

- Q：有限状态集
- Σ：有限输入符号集
- Γ：有限带符号集
- δ：转移函数
- q_0：开始状态
- B：特殊带符：空白符
- F：终态集合

$q_0 \in Q$

$\Sigma \subseteq \Gamma$

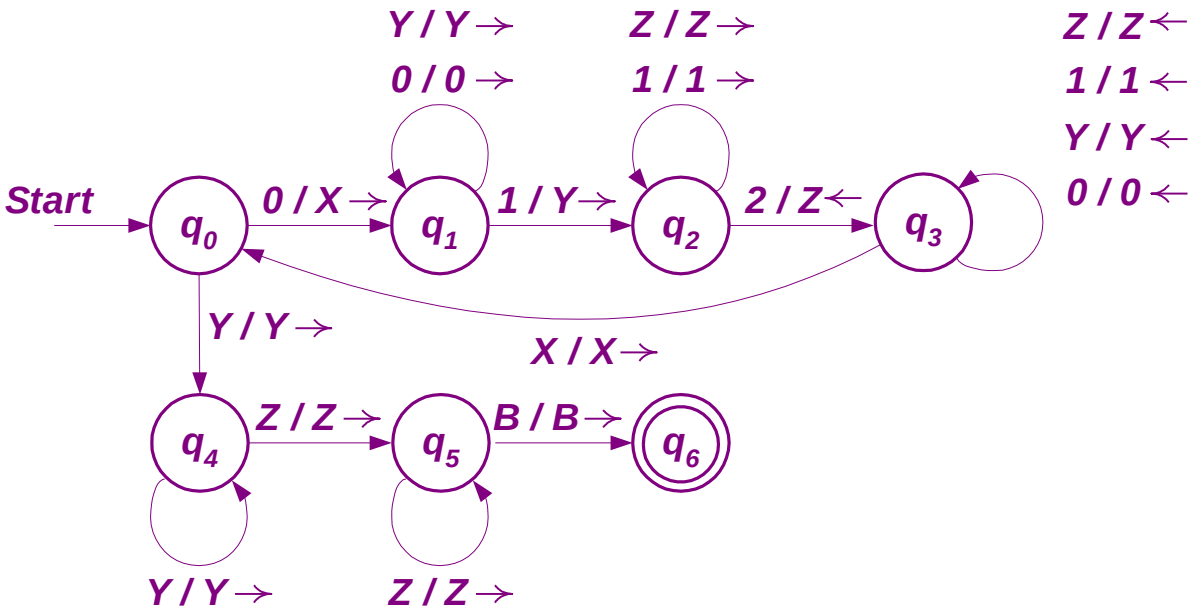
$B \in \Gamma - \Sigma$

$F \subseteq Q$

转移函数为偏函数 $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

图灵机的概念与定义

✧ 举例（续前例）



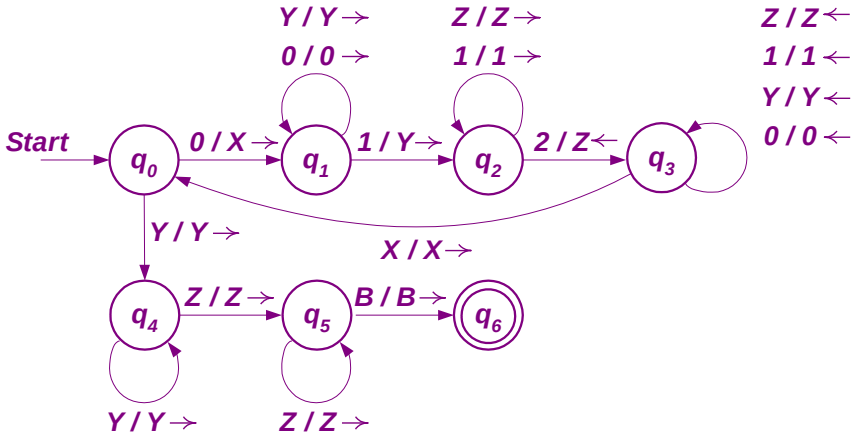
该图灵机的七元组形式为:

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0,1,2\}, \{0,1,2,X,Y,Z,B\}, \delta, q_0, B, \{q_6\}).$$

转移函数可由上图的转移图形式给出.

图灵机的概念与定义

✧ 转移图与转移表



State	Symbol 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_5, Z, R)	—
q_5	—	—	—	—	—	(q_5, Z, R)	(q_6, B, R)
* q_6	—	—	—	—	—	—	—

图灵机的概念与定义

✧ 用*ID*（*instantaneous descriptions*）表达当前格局

图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ 的当前格局用字符串 $X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n$ 表示，称为*ID*，其中

1. $q \in Q$ 为当前 M 的状态；
2. 当前带的头正在扫描 X_i；
3. $X_1X_2\ldots X_n$ 为当前带中最左端的非空白符号与最右端的非空白符号之间的带符号串. 有两个例外，即当带头位于最左端非空白符号的左边时，$X_1X_2\ldots X_n$ 的某个前缀部分为空白符号串；当带头位于最右端非空白符号的右边时，$X_1X_2\ldots X_n$ 的某个后缀部分为空白符号串.

图灵机的概念与定义

✧ 给定图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$，定义 *ID's* 之间的推导关系 $\vdash_M$ 如下：

1. 设 $\delta(q, X_i) = (p, Y, L)$，则有

   $X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n \vdash_M X_1X_2\ldots X_{i-2}pX_{i-1}YX_{i+1}\ldots X_n$，

   但有如下两个例外：

   （1）$i=1$时，　$qX_1X_2\ldots X_n \vdash_M pBYX_2\ldots X_n$，和

   （2）$i=n$及 $Y=B$ 时，　$X_1X_2\ldots X_{n-1}qX_n \vdash_M X_1X_2\ldots X_{n-2}pX_{n-1}$.

2. 设 $\delta(q, X_i) = (p, Y, R)$，则有

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

   但有如下两个例外：

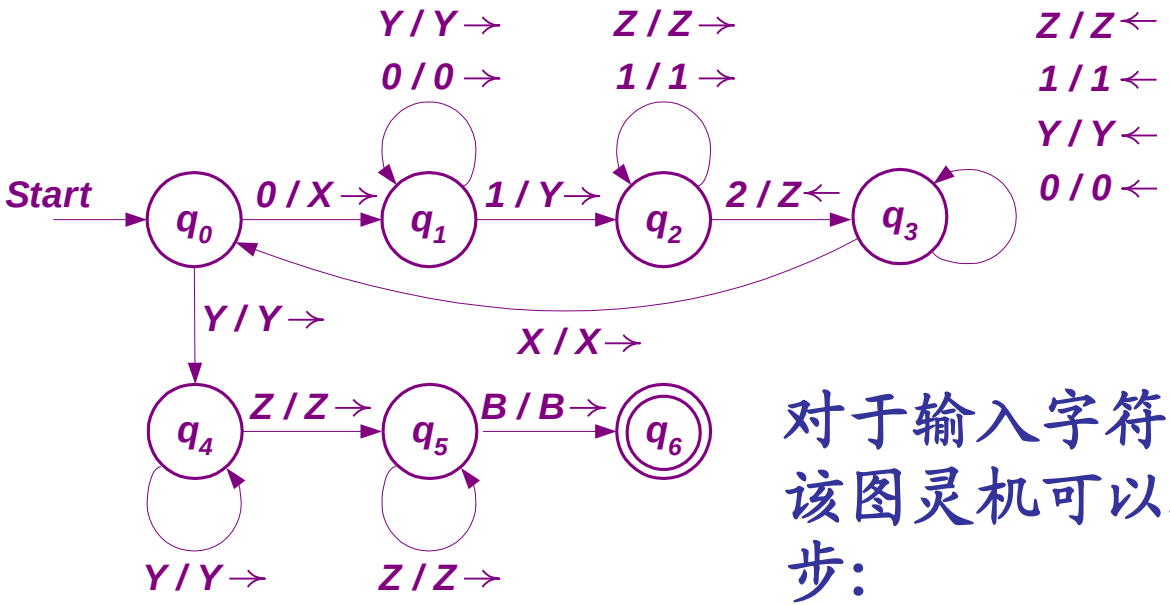
   （1）$i=n$时，　$X_1X_2\ldots X_{n-1}qX_n \vdash_M X_1X_2\ldots X_{n-1}YpB$，和

   （2）$i=1$及 $Y=B$ 时，　$qX_1X_2\ldots X_n \vdash_M pX_2\ldots X_{n-1}X_n$.

✧ 上述关系 $\vdash_M$ 的自反传递闭包记为 $\vdash^*_M$（或 $\vdash^*$）.

图灵机的概念与定义

✧ 举例（续前例）



对于输入字符串001122，该图灵机可以有如下推导步：

$q_0001122 \vdash_M Xq_101122 \vdash_M X0q_11122 \vdash_M X0Yq_2122 \vdash_M X0Y1q_222$
$\vdash_M X0Yq_31Z2 \vdash^*_M q_3X0Y1Z2 \vdash_M Xq_00Y1Z2 \vdash^*_M XXYYZq_22$
$\vdash_M XXYYq_3ZZ \vdash^*_M Xq_3XYYZZ \vdash_M XXq_0YYZZ \vdash^*_M XXYYq_4ZZ$
$\vdash_M XXYYZq_5Z \vdash_M XXYYZZq_5B \vdash_M XXYYZZBq_6B$

递归可枚举语言

- 可以被图灵机接受的语言称为递归可枚举语言（*recursive enumerable languages*）.
- 给定图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$，定义 M 的语言

$$L(M) = \{ w \mid w \in \Sigma^* \wedge \exists p(p \in F \wedge \alpha \in \Gamma^* \wedge \beta \in \Gamma^+ \wedge q_0 w \vdash^*_M \alpha p \beta) \}$$

✧ 图灵机的停机

停机（*halts*）指图灵机不存在下一个移动（*move*）

✧ **结论** 任给图灵机 M，容易构造一个图灵机 M'，使得 $L(M) = L(M')$，并满足：如果$w \in L(M)$，则对于 w，M'接受 w 并一定停机.

由此结论，如果没有特别指出，今后总是假定图灵机到达终态（接受态）后一定停机.

但是，如果$w \notin L(M)$，则对于 w，M 不一定能停机.

递归语言

✧ 递归语言（*recursive languages*）

称语言 *L* 是递归的，当且仅当存在图灵机 *M*，使得 *L* = *L(M)*，且满足:

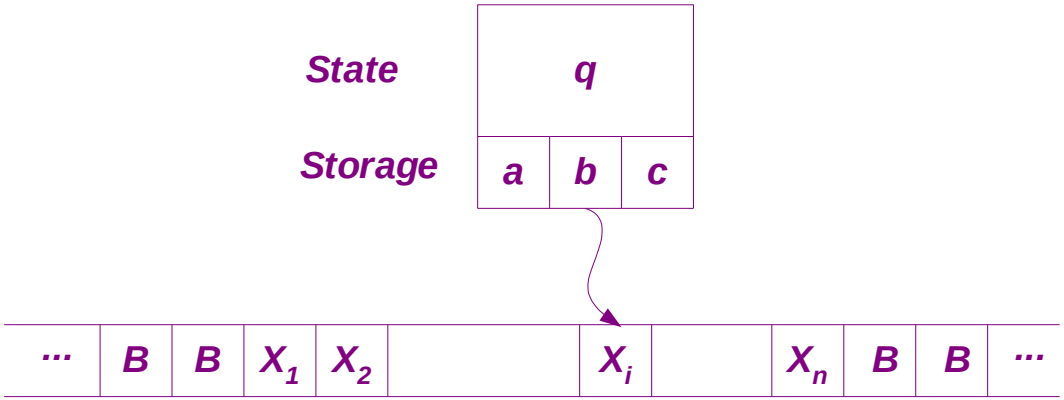
1. 如果$w \in L(M)$，则对于 *w*，*M* 接受 *w*（自然会停机，到达终态）.
2. 如果$w \notin L(M)$，则对于 *w*，*M* 最终也会停机（虽然不能到达终态）.

递归语言对应的问题是可判定的.

（*Church-Turing* 论题）

基本图灵机的几种编程技巧

✧ 利用带存储区的状态（*storage in the state*）



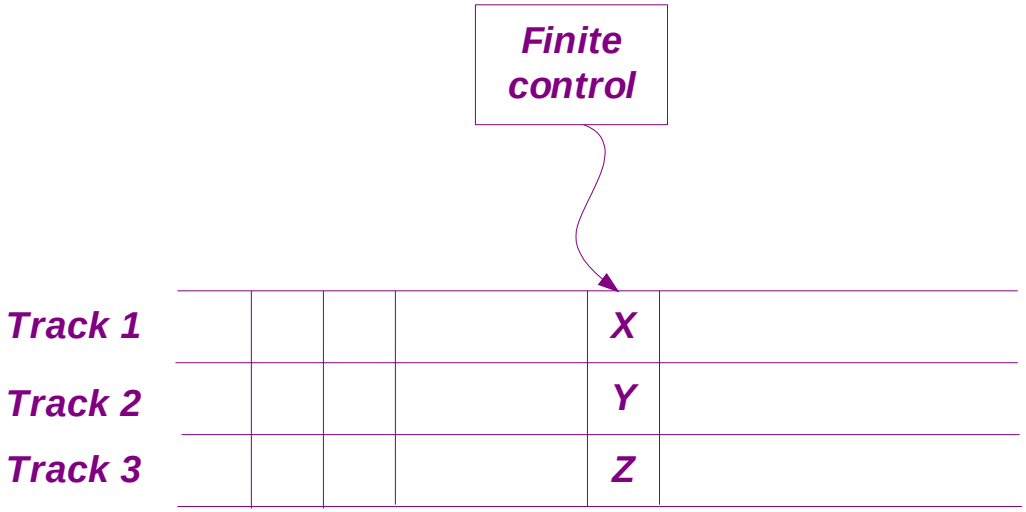
此类图灵机 $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$ 中，状态中可以包含一个具有有限个取值的存储单元，即状态集合为

$$Q=S\times T=\{[q,a]\mid q\in S\wedge a\in T\},$$

其中 $q\in S$ 通常表示控制状态，而 $a\in T$ 通常表示数据元素.

基本图灵机的几种编程技巧

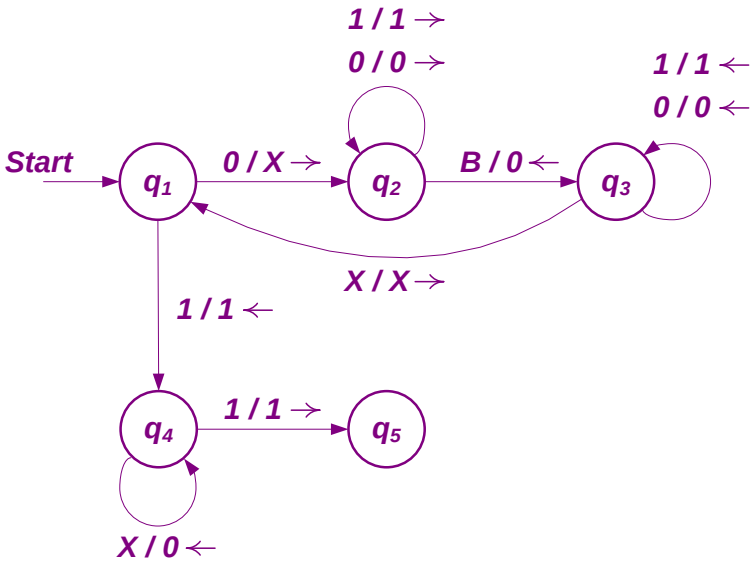
✧ 多道（*multiple tracks*）图灵机

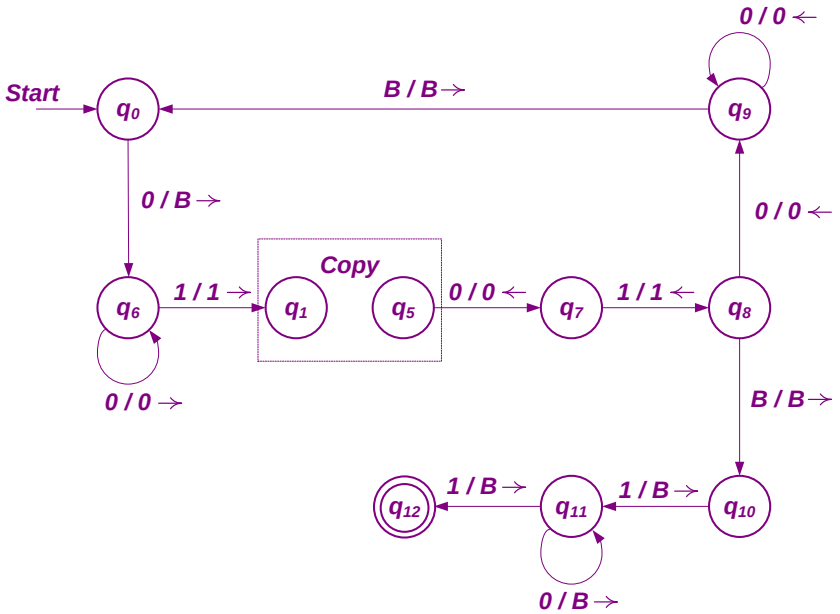


此类图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ 中，带符号可以是元组的形式. 如上图所示的图灵机，带符号的形式为一个三元组.

基本图灵机的几种编程技巧

✧ 子例程（subroutines）的设计





左上图的图灵机表示子例程 *copy*，右上图的图灵机表示可以调用*copy* 的主程序，完成两个正整数的乘法. 初始时，带上的符号串形如 0^m10^n1，而结束时，带上的符号串变为0^{mn}.

对基本图灵机的扩展

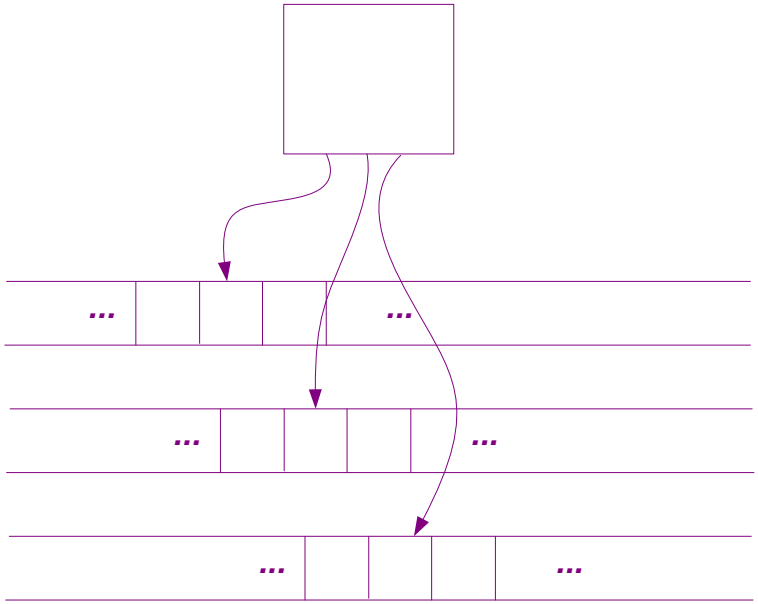
- 多带图灵机（*Multitape Turing Machines*）
- 非确定图灵机（*Nondeterministic Turing Machines*）

对基本图灵机的扩展

✧多带图灵机

– 特点

1. 初始时，输入符号串置于第一条带上；所有带（包括第一条）的其它单元格的符号均为空白符；有限控制处于初态；第一条带的读写头（带头）置于输入符号串的最左端；其余各带的读写头置于任何单元格上.



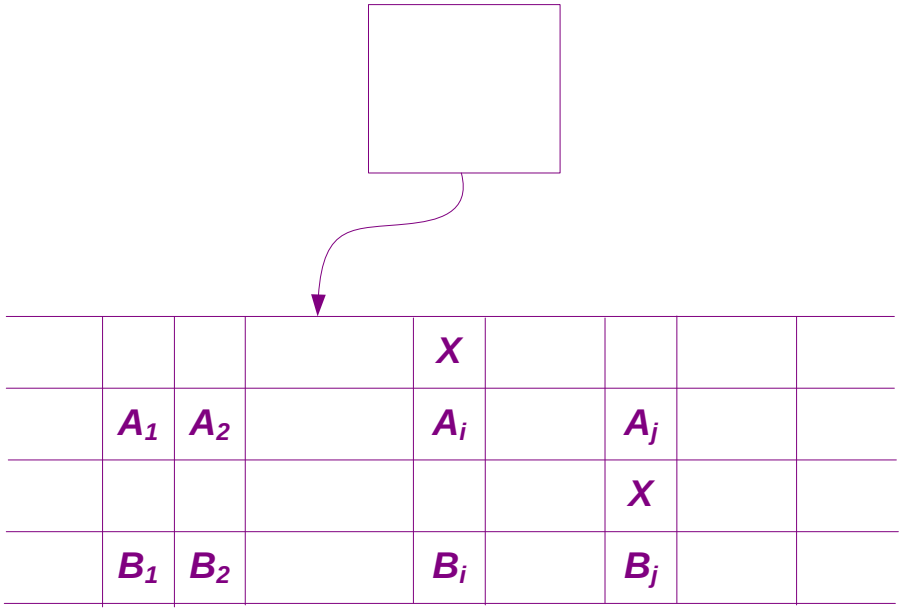
2. 在每一个移动步（*move*）里，控制进入新的状态，每条带上正被扫描的符号被替换为新的带符，每个带头独立地左移一格、右移一格或者不动.

对基本图灵机的扩展

✧多带图灵机

– 语言接受能力与单带图灵机等价

可以采用一个既具有带存储区的状态，又带有多个道的图灵机来模拟. k 个带的图灵机可以用 $2k$ 个道的图灵机来模拟. 下图所示的 4 个道的图灵机用来模拟一个两带的图灵机.

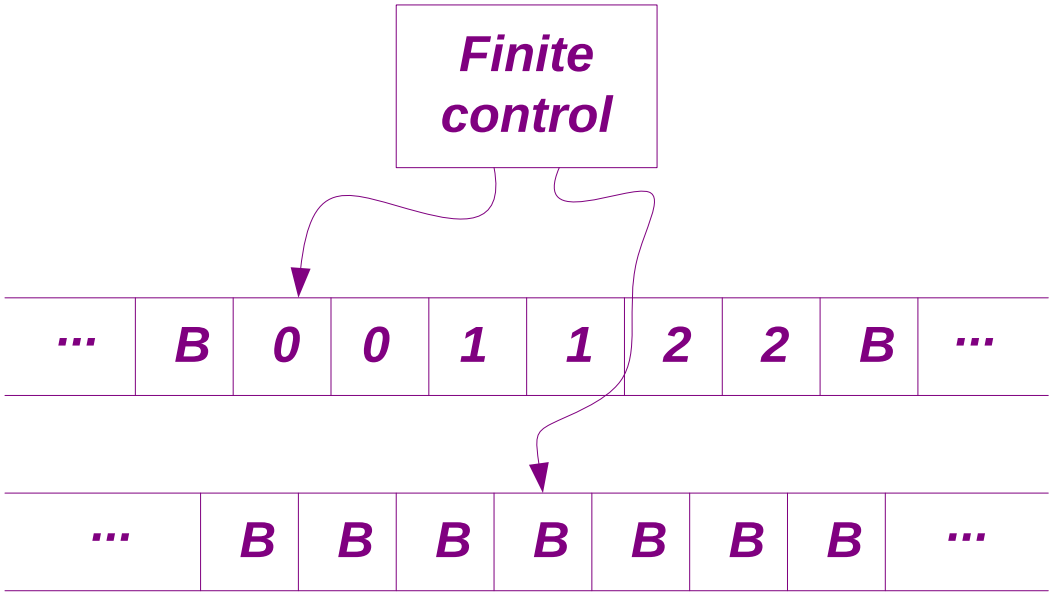


对基本图灵机的扩展

✧多带图灵机

– 举例

一个双带的图灵机如何接受语言 $L = \{ 0^n1^n2^n \mid n \geq 1\}$?



对基本图灵机的扩展

✧ 非确定图灵机

– 下一个动作有多种选择

转移函数可以为 $\delta: Q \times \Gamma \to 2^{Q \times \Gamma \times D}$，其中 Q、Γ 和 D 分别为有限状态集、带符号集和带头的移动方向. 即 $\delta(q,X)$ 为三元组的集合:

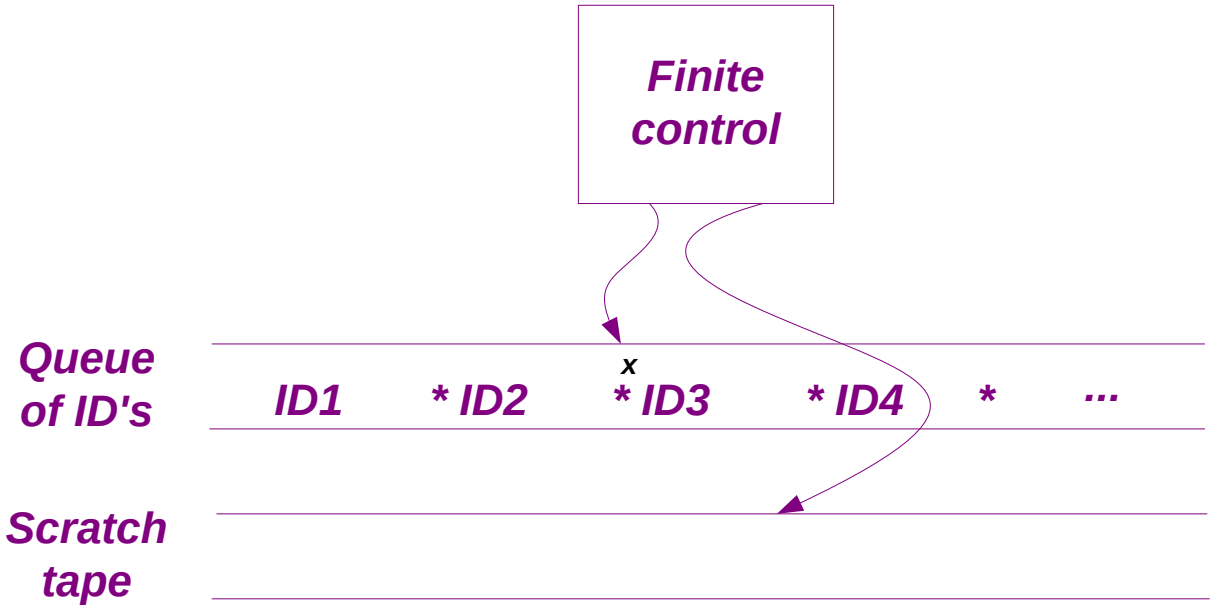
$$\{(q_1,Y_1,D_1),(q_2,Y_2,D_2),\ldots,(q_k,Y_k,D_k)\}$$

对基本图灵机的扩展

✧ 非确定图灵机

– 语言接受能力与（确定的）图灵机等价

可以采用下图所示的多带图灵机来模拟一个非确定图灵机. 主要思想是采用广度优先的方式来模拟非确定图灵机的整个 *ID's* 空间树.



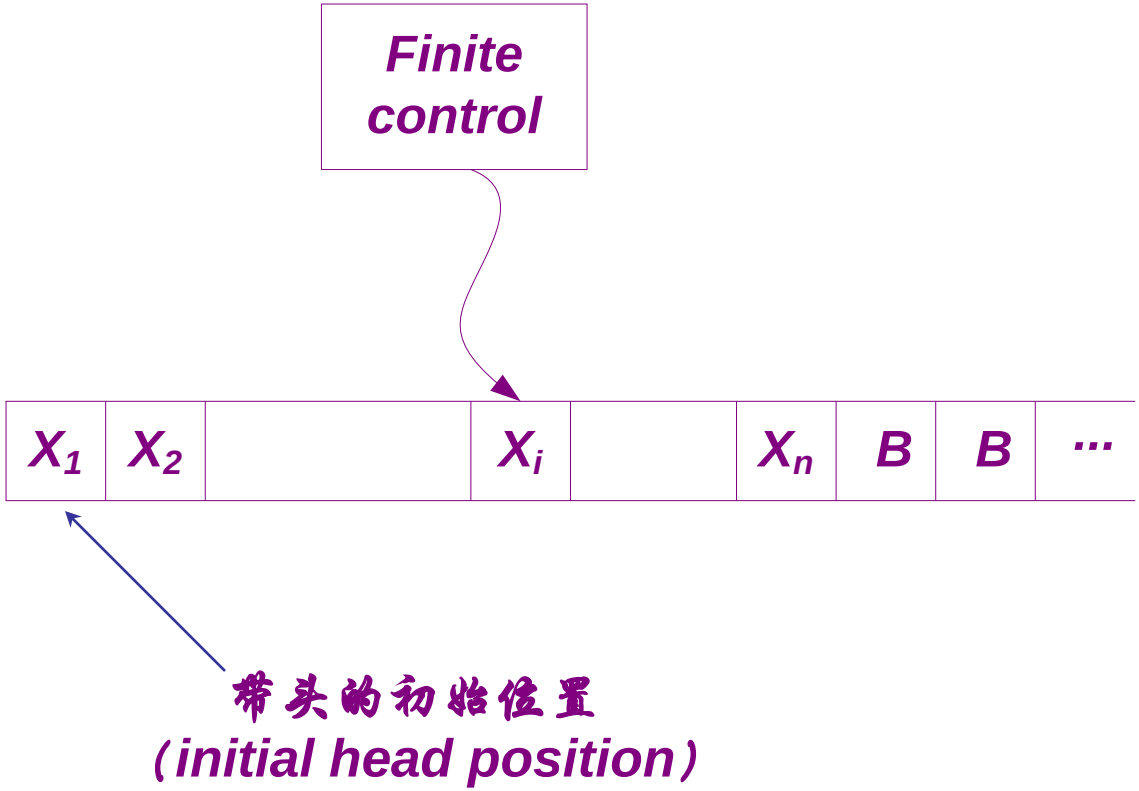
受限的图灵机

- 具有半无穷带（*Semi-infinite Tapes*）的图灵机
- 多栈机（*Multistack Machines*）
- 计数器机（*Counter Machines*）

受限的图灵机

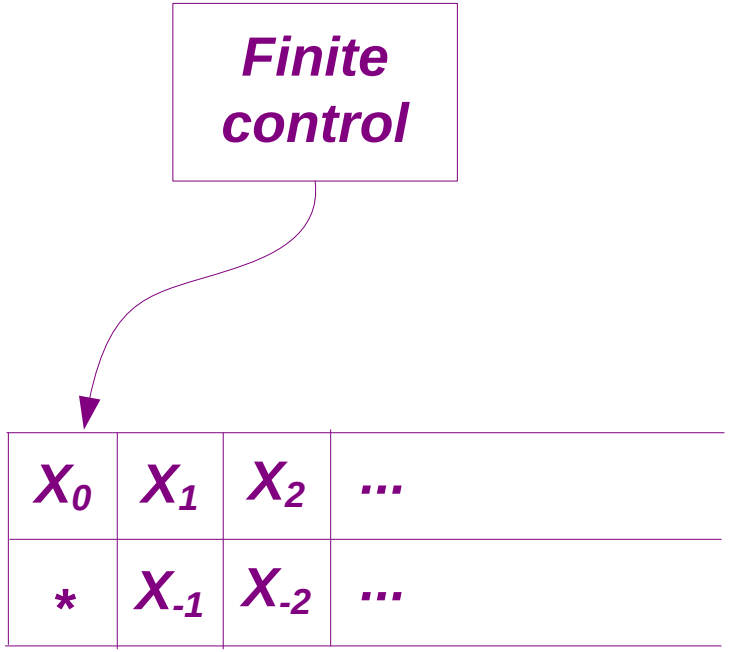
✧ 具有半无穷带的图灵机

– 带头的初始位置左部没有单元格，带头移动受限于从带头初始位置开始向右无限延伸的范围.



受限的图灵机

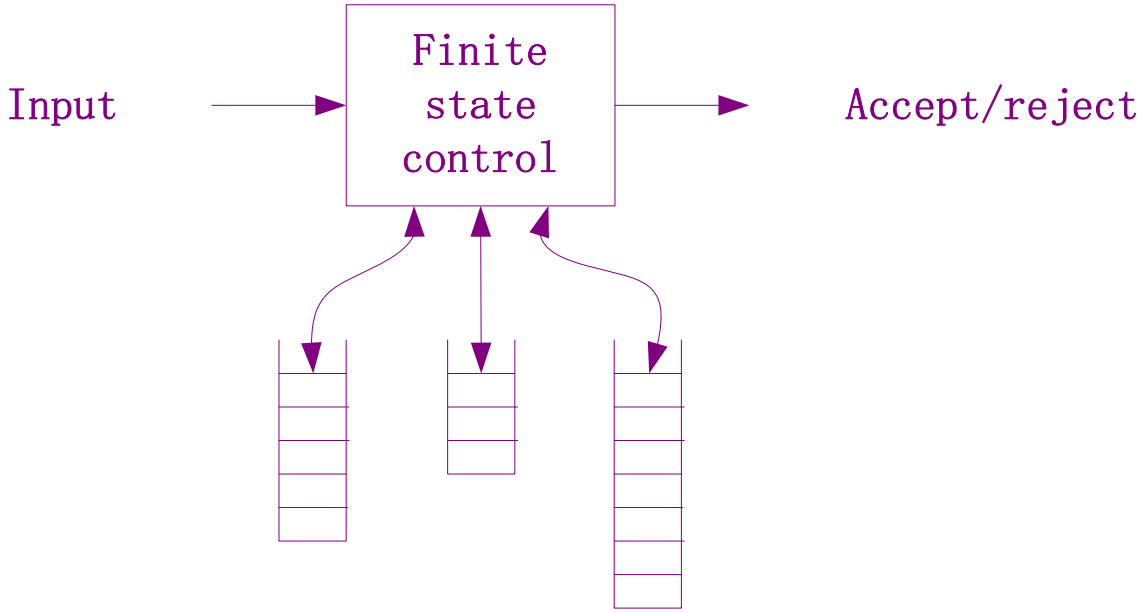
- 具有半无穷带的图灵机
  - 可以用一个双道的半无穷带图灵机模拟具有双向无穷带的基本图灵机.



受限的图灵机

✧ 多栈机

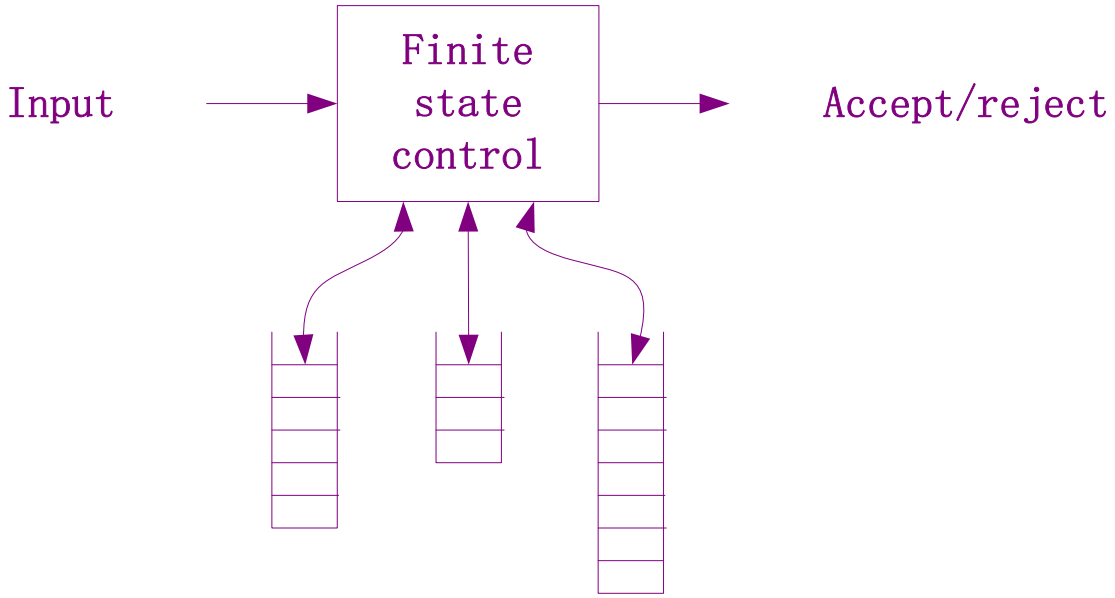
– 具有多个下推栈的 PDA，包含 k 个下推栈的一步转移可表达为 $(p, \gamma_1, \gamma_2, \ldots, \gamma_k) \in \delta(q, a, X_1, X_2, \ldots, X_k)$.
下图示意带有 3 个下推栈的PDA.



受限的图灵机

✧ 多栈机

– 利用一个多带图灵机很容易模拟多栈机. 例如，下图所示的多栈机可以采用 *4* 条带的多带图灵机模拟： 一条带用于扫描输入，另 *3* 条带模拟下推栈.

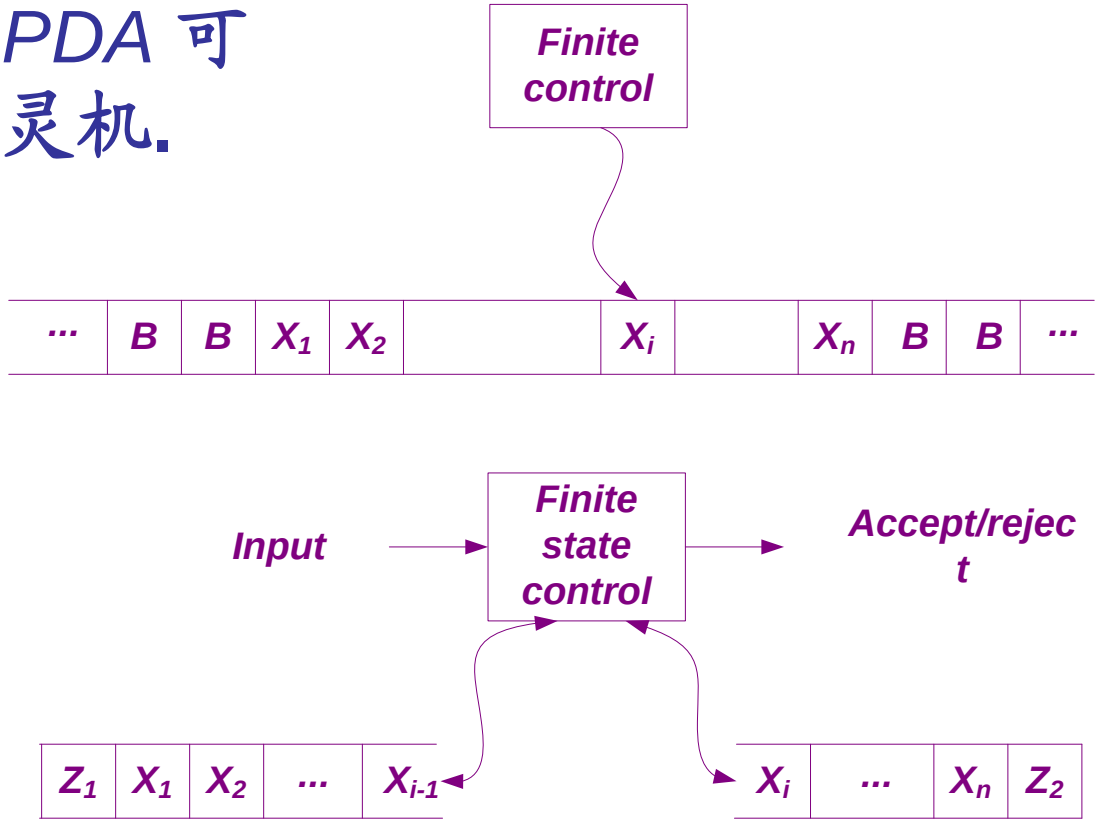


受限的图灵机

✧ 多栈机

– 利用一个双栈 PDA 可以模拟基本图灵机.

如右图所示，第一个下推栈模拟当前带头位置左边的单元格，而第二个下推栈模拟当前带头位置右边（包含当前）的单元格.



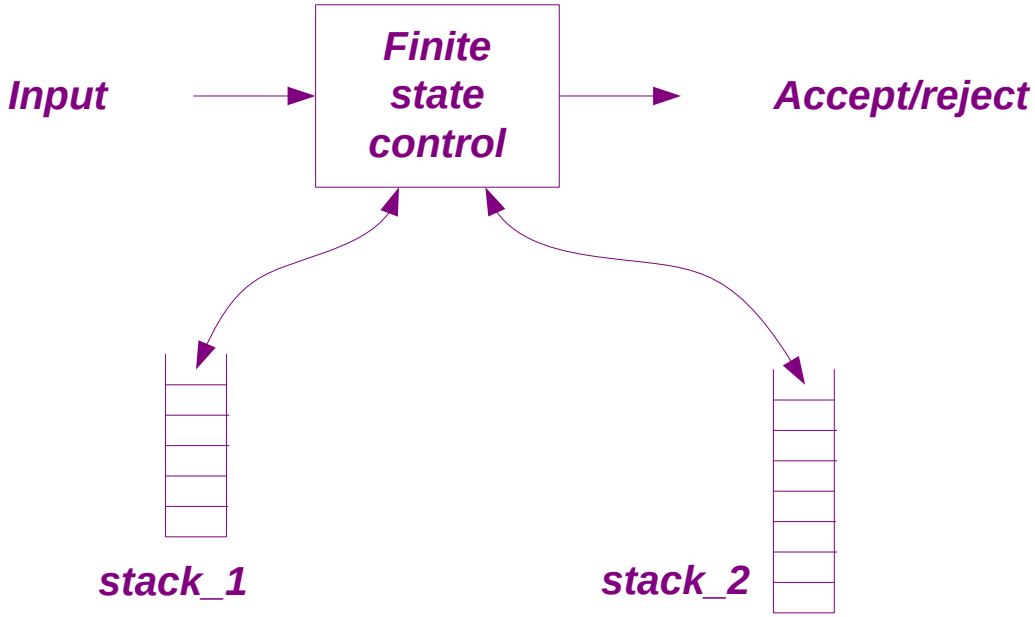
受限的图灵机

✧ 多栈机

– 举例

如何使用一个双栈的 PDA 接受语言

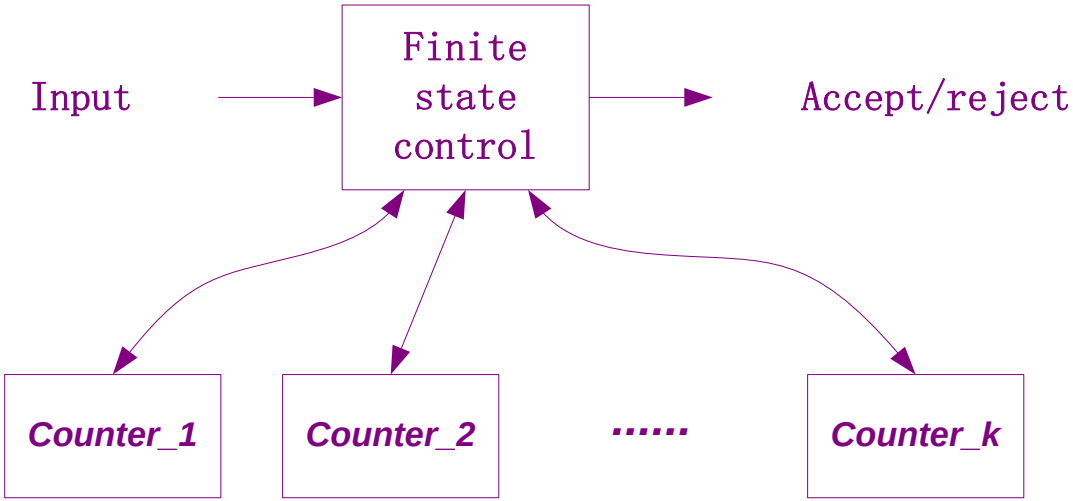
$$L = \{ 0^n 1^n 2^n \mid n \geq 1 \}?$$



受限的图灵机

✧ 计数器机

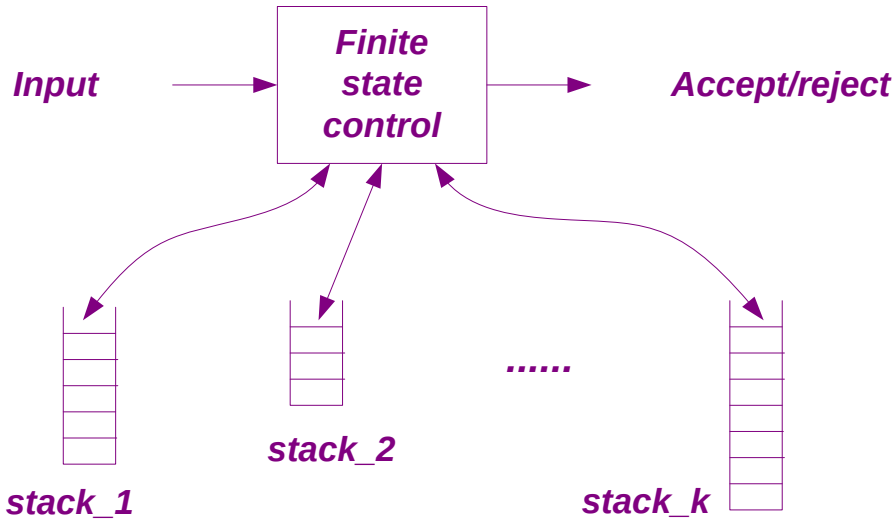
– 将多栈机的多个下推栈替换为多个计数器，每个计数器存放一个非负整数，控制只知道每个计数器的值是0或是非 0．计数器机根据当前的状态，下一个输入符号以及每个计数器的当前值是否为 0 来决定下一步动作：改变状态，计数器加 1 或减 1，但不允许对当前值为 0 的计数器减 1.



受限的图灵机

✧ 计数器机

– 可以认为，计数器机是一个特殊的多栈机：只有两个栈符号，一个为栈底标志 Z_0，另一个为 X；每个栈都初始化为 Z_0，Z_0 只能被替换为 X^iZ_0（$i \geq 0$），X 只能被替换为 X^i（$i \geq 0$）. 栈顶为 Z_0 和 X 分别相当于计数器的值是0 或是非 0 . X 入栈相当于计数器加 1，X 出栈相当于计数器减 1，Z_0 不会出栈相当于计数器为 0 时不能减 1.



受限的图灵机

✧ 计数器机

- 由于上述特殊的多栈机相当于计数器机，因此计数器机所接受的语言都是递归可枚举的；反过来，所有递归可枚举语言是否都存在相应的计数器机？关于计数器机所能接受的语言有如下结论:

- 结论 具有一个计数器的计数器机的语言接受能力相当于确定的下推自动机.

- 结论 具有两个（或以上）计数器的计数器机的语言接受能力相当于图灵机，即可以接受递归可枚举语言.

  证明 分两步：（1）任何递归可枚举语言可以被具有三个计数器的计数器机接受；（2）任何具有三个计数器的计数器机可以用一个具有两个计数器的计数器机来模拟.

受限的图灵机

✧ 计数器机

– **结论** 任何递归可枚举语言可以被具有三个计数器的计数器机接受.

证明思路 用具有三个计数器的计数器机模拟任何两个下推栈的多栈机.

假定共有 *r-1* 个堆栈符号，并将它们分别用整数 *1* 到 *r-1* 表示. 这样，一个栈 $X_1X_2\ldots X_n$ 可编码为

$$X_n r^{n-1} + X_{n-1} r^{n-2} + \ldots X_2 r + X_1 \ .$$

可以用两个计数器表示两个下推栈，第三个计数器用于辅助完成关于栈的操作. 前者初始化为开始栈符对应的整数，后者初始化为 *0*. 操作 *pop* 相当于上述整数除以 *r* 取整（即弹出X_1），操作 *push(X)* 相当于乘以 *r* 再加上*X*，将栈顶元素 *X* 替换为*Y* 相当于先减去 *X* 再加上 *Y*. 乘和除需借助于第三个计数器来完成.

受限的图灵机

✧ 计数器机

– **结论** 任何具有三个计数器的计数器机可以用一个具有两个计数器的计数器机来模拟.

证明思路 将任何三个计数器 i, j, k 用整数 $m=2^i3^j5^k$ 表示，存放于一个计数器，另外一个计数器用于辅助完成相应于原先 i, j, k 计数器上的操作.

主要模拟 i, j, k 计数器上的三种操作:

(1) i, j, k 加1 i 加 1 相当于上述计数器 m 乘以 2， j 加 1 相当于m 乘以 3， k 加1 相当于m 乘以 5;

(2) i, j, k 减1 i 减 1 相当于上述计数器 m 除以 2， j 减 1 相当于m 除以 3， k 减1 相当于m 除以 5;

(3) 判别 i, j, k 是否为 0 相当于分别判别 m 能否被 2, 3 和 5 整除. 能整除则不为 0 ，反之则为 0.

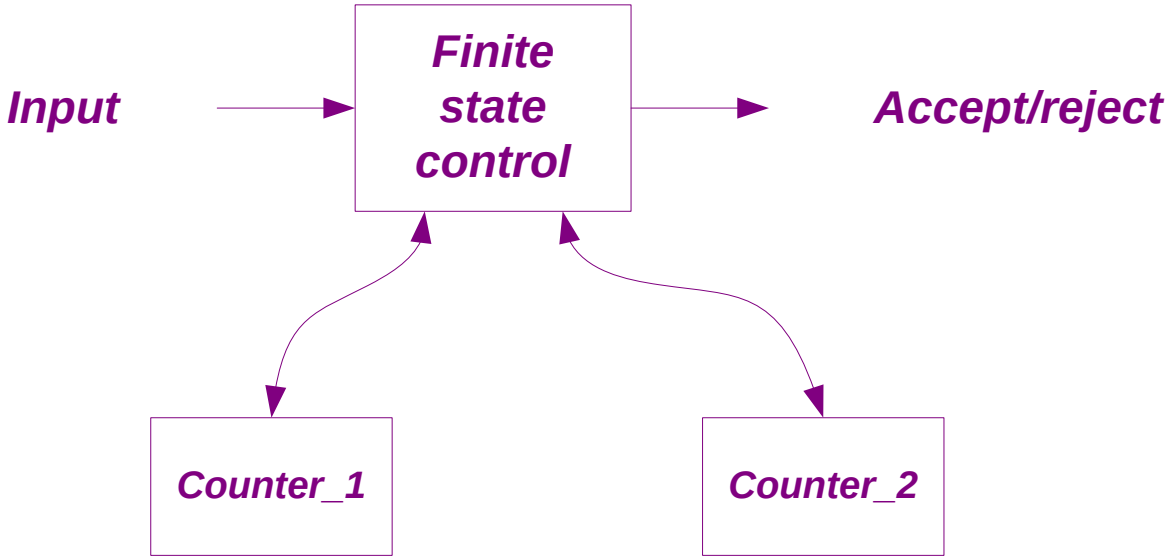
受限的图灵机

✧ 计数器机

– 举例

如何使用一个双计数器的计数器机接受语言

$$L = \{ 0^n 1^n 2^n \mid n \geq 1 \}?$$



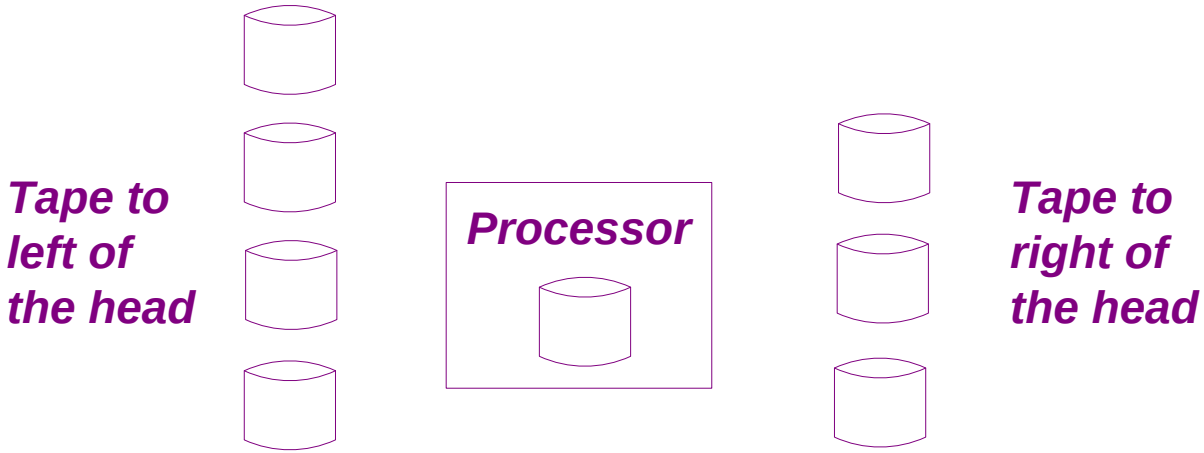
图灵机与计算机

- 以普通计算机模拟图灵机
- 以多带图灵机模拟普通计算机

图灵机与计算机

✧以普通计算机模拟图灵机

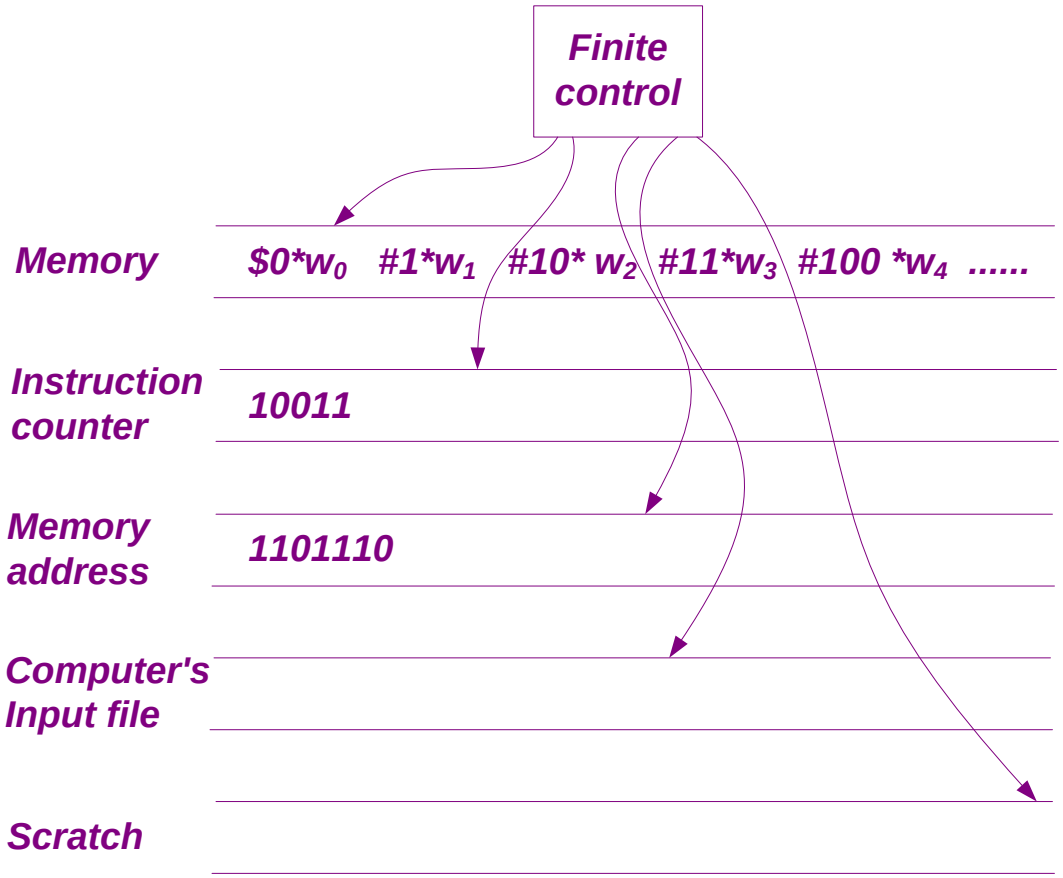
– 采用适当的数据结构（如转移表）不难编制普通的计算机程序实现图灵机的有限状态控制机制. 存在问题的是如何模拟无限延伸的带，因为普通计算机的存储空间（包括各个级别的存储器）是有限的. 但是，可以假想一种可以无限扩充存储量的存储系统. 实际上，可装卸的 外存系统并不严格规定存储量的上限，而且并非所有信息都需要在线存储.



图灵机与计算机

✧ 以多带图灵机模拟普通计算机

– 可以用多带图灵机模拟典型的存储程序式计算机，参见以下示意图. 必要时，可增加更多的带.



课后练习

✧ 必做题:

- !Ex.8.2.2(c)
- Ex.8.2.3
- Ex.8.2.5(b)

✧ 思考题:

That's all for today.

Thank You