

CH13、有限狀態機

有限狀態機

目錄：

- 13-1 有限狀態機
 - 輸入/輸出、狀態集、轉換函數、輸出函數
 - Mealy Model
- 13-2 有限狀態機的簡化
 - 狀態等價、將態簡化演算法(*Moore State Reduction Algorithm*)
- 13-3 語言學
 - 符號集、空字串
 - 語言、串連、正包、Kleene 包
- 13-4 文法
 - 文法、直接推導出、推導出、推導過程
 - 生成的語言
 - 型態 0~型態 3
- 13-5 自動狀態機
 - 自動狀態機
 - 接受狀態、拒絕狀態、接受字串、拒絕字串
 - 有限狀態語言
 - Pumping 引理
- 13-6 非決定性自動狀態機
 - Non-deterministic Finite State Automata, NFSA
- 13-7 正規化表示法
 - 正規化表示法、正規集
 - Kleene's 定理
- 13-8 Turing 機
 - (略)

13.1 有限狀態機

定義：

一個 Finite State Machine(FSM)

$M = [I, o, S, v, w]$

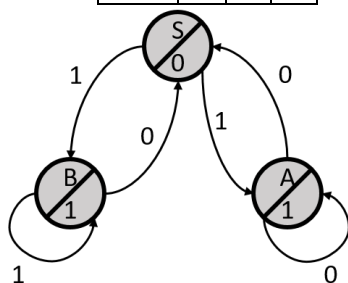
I = Input Set , o = Output Set , S = State Set, $|S| < \infty$

$v = S \times I \rightarrow S$ (狀態轉換函數 *State Transition Function*) , w = Output Function

w 分為兩種：

1. Moore Mode

start→	v		w
	0	1	
S	A	B	0
A	A	S	1
B	S	B	1



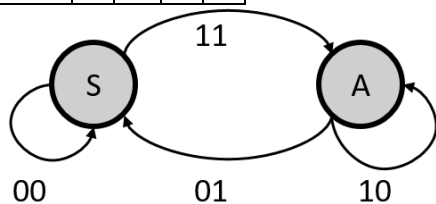
Input : 0011

$S \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{1} S \xrightarrow{1} B$

Output : 01101

2. Mearly Mode(較常見)

State	v		w	
	0	1	0	1
S	S	A	0	1
A	S	A	1	0

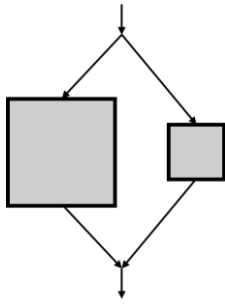


Input : 0011

$S \xrightarrow{00} S \xrightarrow{00} S \xrightarrow{11} A \xrightarrow{10} A$

Output : 0010

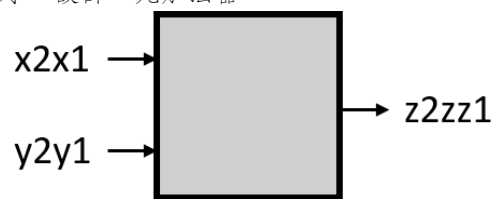
無 Hard Disk



*表示法：

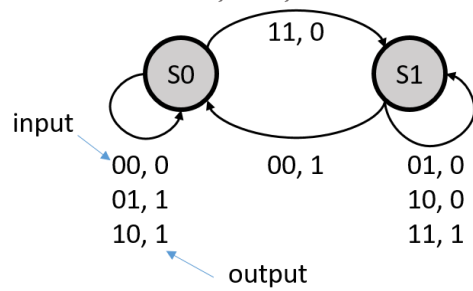
1. State Table
2. State Diagram

例：設計二元加法器



$I = \{00, 01, 10, 11\}, o = \{0, 1\}$

$S = \{S_0, S_1\}$ //No carry, carry



13.2 有限狀態機簡化

定義：

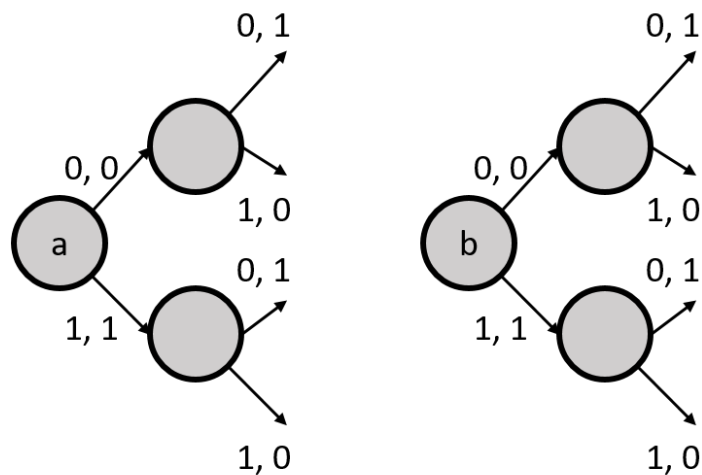
$M = [I, o, S, v, w] = \text{FSM}$

在 S 上定義 Equivalent Relation：

E_k ：k-equivalent

E ：State-Equivalent

1. $aE_kb \Leftrightarrow$ 在 input 長度為 k 時， a, b 具相同 Output



$0 \rightarrow 0, 00 \rightarrow 01, 1 \rightarrow 1 \quad 0 \rightarrow 0, 00 \rightarrow 00, 1 \rightarrow 1$

a, b 為 1 等價，非 2 等價

2. $aEb \Leftrightarrow a, b$ 在任意 Input 具相同 Output

Note：

1. $E_{k+1} \subseteq E_k$

$\forall E_{k+1}$ 都符合 E_k ，但 E_k 未必符合 E_{k+1}

2. 對應分割 P_k, P_{k+1} ，則 P_{k+1} 為 P_k 之加細分割(關係愈小，切得愈細)

例：

	v		w	
	0	1	0	1
1	5	7	1	0
2	7	2	1	0
3	6	1	1	0
4	3	4	0	0
5	3	5	0	0
6	2	7	1	0
7	4	1	1	0

1. 求 Reduced(最簡) Machine？

2. 求將 State 3 及 6 分開之最短字串？

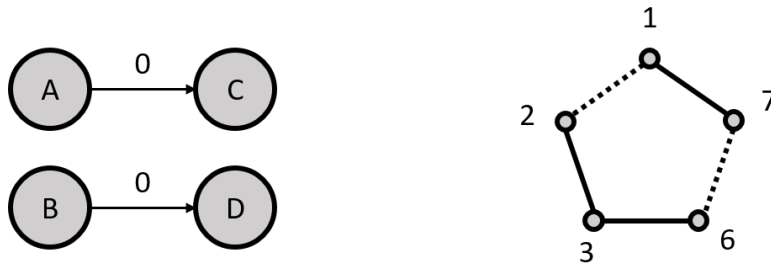
1. 步驟：

(1) $P1 = \{\{1, 2, 3, 6, 7\}, \{4, 5\}\}$

1 等價，看 $Output(1, 0)$ 與 $(0, 0)$

(2) $P2$ ：看 State Transition：到 1 等價的 State 則 Output 相同

若 CD 為 1 等價，則 A, B 為 2 等價 \rightarrow 輸入 1 也是 2 等價的話，才可將 A, B 歸到 2 等價



其餘不用看，因為 Equivalent Relation 有 Transitive，故：

$P2 = \{\{1, 7\}, \{2, 3, 6\}, \{4, 5\}\}$

(3) $P3 = \{\{1, 7\}, \{2\}, \{3, 6\}, \{4, 5\}\}$

(4) $P4 = \{\{1, 7\}, \{2\}, \{3\}, \{6\}, \{4, 5\}\}$ // 無法再細切

$\hookrightarrow A = \{1, 7\}, B = \{2\}, C = \{3\}, D = \{4, 5\}, E = \{6\}$

\Rightarrow

	v		w	
State			0	1
A	E	A	1	0
B	A	B	1	0
C	D	A	1	0
D	B	A	1	0
E	C	E	0	0

2. 回推步驟

(1) $\{3\}, \{6\}$ 回看為 $\{6\}, \{2\}$ ，輸出為 0, 0

(2) $\{6\}, \{2\}$ 回看為 $\{7\}, \{2\}$ ，輸出為 1, 1

(3) $\{7\}, \{2\}$ 回看為 $\{4\}, \{7\}$ ，輸出為 0, 0

(4) $\{4\}, \{7\}$ 回看輸出為 0, 1 \rightarrow 0100

13.3 語言學

定義：

Σ = Symbol Set

$\Sigma^n = \{a_1 a_2 \dots a_n \mid a_1, \dots, a_n \in \Sigma\}$

$\alpha \in \Sigma^n, |\alpha| = n$ (長度)

$\Sigma^+ = \bigcup_1 \Sigma^n$

$\Sigma^* = \bigcup_0 \Sigma^n = \Sigma^+ \cup \Sigma^0$

$\Sigma^0 = \{\lambda\}$, λ 為 Empty String

$A \subseteq \Sigma^*$, 稱 A 為 Language over Σ

$A, B \subseteq \Sigma^*$

1. $A \cdot B = \{\alpha \cdot \beta \mid \alpha \in A, \beta \in B\}$

2. $A+B = A \vee B = A|B = A \cup B$

3. $A^2 = AA$

$A^n = AA \dots A$

4. $A^+ = \bigcup_1 A^n, A^* = A^+ \cup \{\lambda\}$

例：True/False

$(A^*)^2 = (A^2)^*$

$A = \{a\}, A^2 = \{aa\}, (A^2)^* = \{\lambda, a^2, a^4, \dots\}$

$A^* = \{\lambda, a, a^2, a^3, \dots\}$

$(A^*)^2 = \{\lambda, a, a^2, a^3, \dots\} \Rightarrow \text{False}$

例： $(A^*)^* = A^*$

例(96 台大)： $A = \{1, 10\}$ ，下列何者在 A^* 中？

1. λ
2. 1101
3. 1110110
4. 10110
5. 110011

True : 1, 2, 3, 4 : False : 5

例(97 中原)：遞迴定義下列 Language

1. 開頭 1，長度偶數之 Bit String ?
1000 1001 1010 1011 1100 1101 1110 1111 ...
Base Case : 10, 11 $\in S$ (所求)
Recursive Step :
if $x \in S, x00, x01, x10, x11 \in S$
2. 1 的個數比 0 的個數多之 Bit String
Base Case : 1 $\in S$ (所求)
Recursive Step :
if $x, y \in S$, then $0xy, x0y, xy0, xy \in S$

13.4 文法

定義：

一個 Grammar $G = (S, N, T, P)$

S = Starting Symbol

N = Nonterminals Set (大寫)

T = Terminals Set (小寫)

P = Production Rule (每 1 條具型式 $\alpha \rightarrow \beta$)

$\alpha \in (NUT)^* - T^*$ (不可全小寫)

$\beta \in (NUT)^*$

分成四種 Type：

1. Type 0：無限制

2. Type 1(又稱為 *Context-Sensitive Grammar*)： $\delta A w \rightarrow n r \Phi$, $\delta, w, n, \Phi \in (NUT)^*$, $A \in N$, $r \in (NUT)^* - \lambda$

3. Type 2(又稱為 *Context-Free Grammar*)： $A \rightarrow B$, $A \in N$, $B \in (NUT)^*$

4. Type 3(又稱為 *Regular Grammar*)： $A \rightarrow a$, $A \rightarrow aB$, $a \rightarrow \lambda$, $A, B \in N$, $a \in T$

其中： G 可導出之所有字串集合記作 $L(G)$

例：

G ：

$S \rightarrow 0A$

$S \rightarrow 1A$

$A \rightarrow 0B$

$B \rightarrow 1$

$B \rightarrow 1A$

$A \rightarrow 0B \rightarrow 01A \rightarrow 010B \rightarrow 0101A \rightarrow 01010B \rightarrow 010101$

$\Rightarrow L(G) = \{0(01)^k \mid k \geq 1\} \cup \{1(01)^k \mid k \geq 1\}$

例(96 中央)： $L : \{w \in \{0, 1\}^* \mid w = wR\}$ ，求 $G \ni L(G) = L$

G ：

$S \rightarrow 0S0$

$S \rightarrow 1S1$

$S \rightarrow 1$

$S \rightarrow 0$

$S \rightarrow \lambda$

例(94 中正)： $L : \{0^m 1^{m+n} \mid m, n \geq 0\}$ ，求 $G \ni L(G) = L$

G ：

$S \rightarrow AB$

$A \rightarrow 0A1$

$B \rightarrow 1B0$

$A \rightarrow \lambda$

$B \rightarrow \lambda$

或寫成

$A \rightarrow 0A1 \mid \lambda$

$B \rightarrow 1B0 \mid \lambda$

13.5 自動狀態機

定義

一個 Finite State Automaton(FSA)

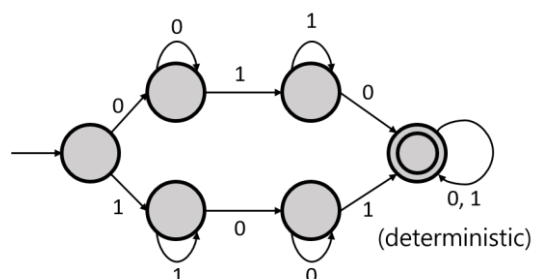
$M = [S_0, I, S, v, A]$

S_0 =Starting State ; I, S, v 同 FSM

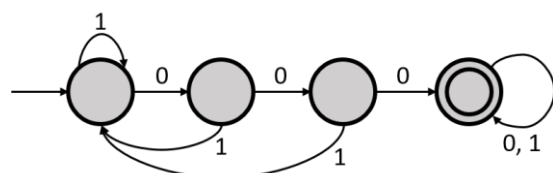
$A \subseteq S$

1. $a \in A$ ，稱 a 為 Accept State，記作 
- $a \notin A$ ，稱 a 為 Reject State，記作 
2. $\alpha \in I^*$ ，在 Input α 之下，若停在 Accept State 稱 M accept α
- $\alpha \in I^*$ ，在 Input α 之下，若停在 Reject State 稱 M reject α
3. 將所有被接受之字串收集成 L ，稱 M 認知 L

例(99 長庚)： $L = \{w \in \{0, 1\}^* \mid w \text{ 有 substring } 01 \text{ 及 } 10\}$ ，求 M 認知 L



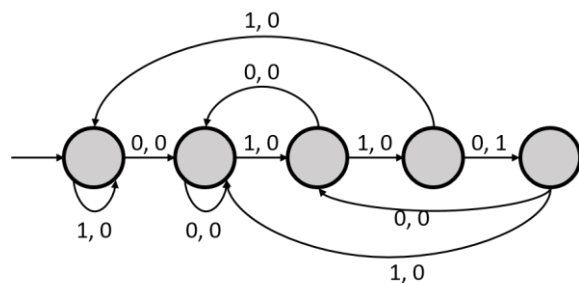
例(99 市北)： $\{\alpha \in \{0, 1\}^* \mid \alpha \text{ 含 3 個連續 } 0\}$



例(93 師大)：設計一個 FSM 認知所有含 Pattern 0110 之字串(允許 overlap)

Input : 11 0110 1 0110 110 10

Output : 000001000100100 (1 為有出現)→比 FSA 更強，可知有幾個、和在這裡



以 FSM 設計 FSA 更有彈性+強大

Note :

可被 FSA 認知之語言稱為 Finite State Language (FSL)

例(12 個) : 證 $L = \{akbk \mid k \geq 1\}$ is not FSL

證明 :

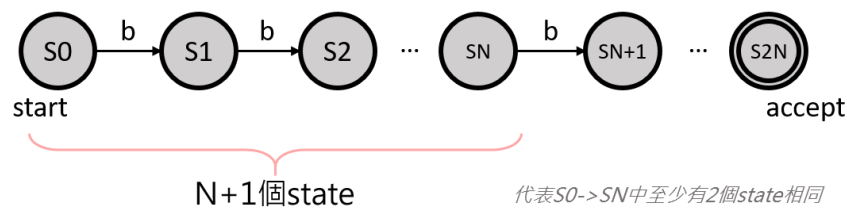
by 矛盾證法

設 L 為 FSL

則 \exists 一個 FSM : M 認知 L

令 M 之 State 個數為 N

取 $\alpha = a^N b^N \in L$, 放入 M 中



By pumping lemma

α 可拆成 u, v, w

其中 $|v| \geq 1$ 且 v 中只含 $a \exists u v^k w \in L, \forall k \geq 0$

取 $uv^0w = uw \in L \rightarrow \leftarrow$

有『記憶性』的 FSM/FSA 都沒辦法處理，例： $\{a^p \mid p : \text{Prime}\}$

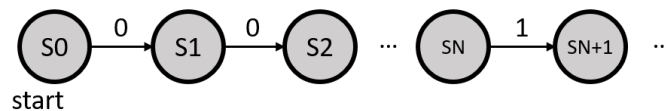
例(98 北科) : $L : \{w \in \{0, 1\}^* \mid w = w^R\}$

by 矛盾證法

設有一 FSA accept L

FSA 存在 N 個 State

取 $\alpha = 0^N 11 0^N$



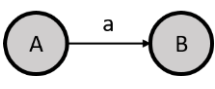
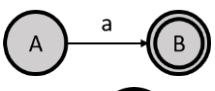
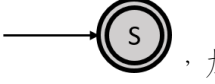
by pumping lemma α 可拆成 uvw

其中 $|v| \geq 1$ 且只含 $0 \exists uv^i w \in L, \forall i \geq 0$

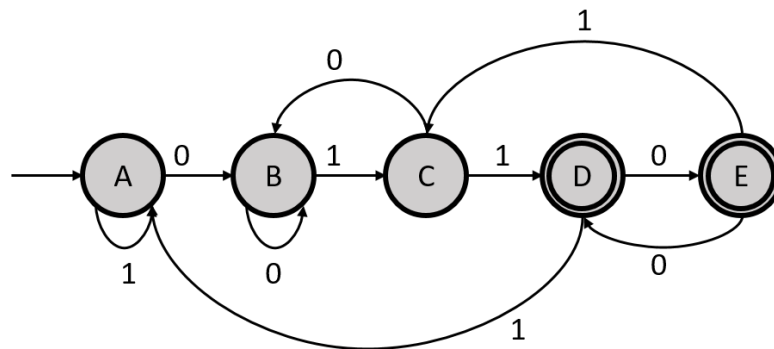
$\Rightarrow uv^0w \in L \Rightarrow uw \in L \rightarrow \leftarrow$

定理 :

1. FSA 轉成 Regular Grammer (Type 3)
2. 反之亦可

1.  , 加入 $A \rightarrow aB$
2.  , 加入 $A \rightarrow aB \mid a$ //到B繼續走
3.  , 加入 $S \rightarrow \lambda$

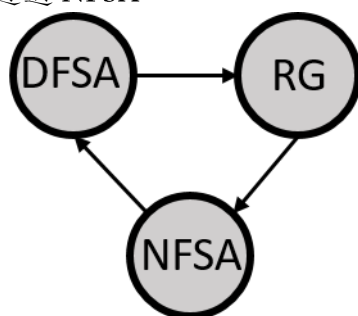
例(94 清大)：請轉成 Type 3 Grammer



$A \rightarrow 0B \mid 1A$
 $B \rightarrow 0B \mid 1C$
 $C \rightarrow 1 \mid 1D \mid 0B$
 $D \rightarrow 0 \mid 0E \mid 1A$
 $E \rightarrow 0 \mid 0D \mid 1C$

如何將 Regular Grammer 轉回 DFSA(DFA)呢？

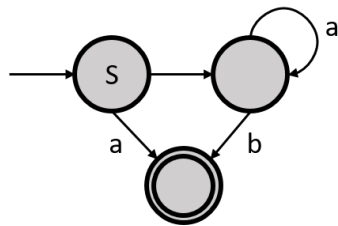
透過 NFSA



13.6 非決定性自動狀態機

$S \rightarrow 0A \mid 0$

$B \rightarrow 1A$



定義：

Nondeterministic FSA (NFSA)

1. 一個 State 在同樣 Input 下可有 ≥ 1 條路，有任一個結果是 Accept 即算 NFSA accept input
2. 所有狀態不用畫出所有 Input 之結果，若無此 Input 之路，即 NFSA reject input

例：將此 Grammer 轉成 NFSA

$S \rightarrow 1B$

$S \rightarrow 0$

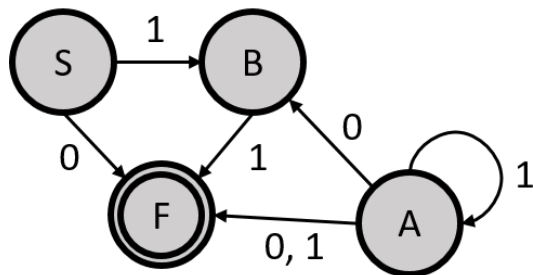
$A \rightarrow 1A$

$A \rightarrow 0B$

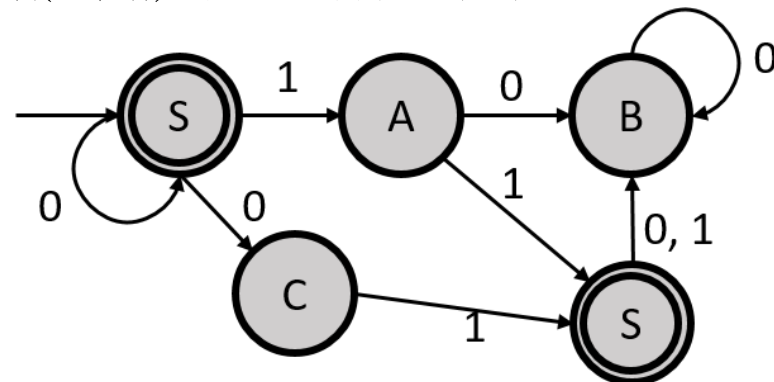
$A \rightarrow 1$

$A \rightarrow 0$

$B \rightarrow 1$



例(99 台科)：將此 NFSA 轉成 DFSA(DFA)



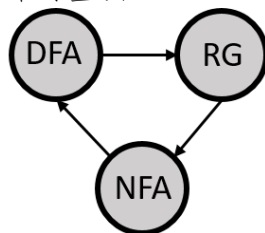
	v		w
State	0	1	
S	S, C	A	1
A	B	D	0
B	B		0
C		D	0
D	B	B	1

	v		w
State	0	1	
{S}	{S, C}	{A}	1
{S, C}	{S, C}	{A, D}	1
{A}	{B}	{D}	0
{A, D}	{B}	{B, D}	1
{B}	{B}	\varnothing	0
{D}	{B}	{B}	1
{B, D}	{B}	{B}	1
\varnothing	\varnothing	\varnothing	0

看State

只要看S 開始走『有經過』的集合點。例：沒有{C}所以 State 不用列{C}

即可畫出 DFA



三者能力一樣

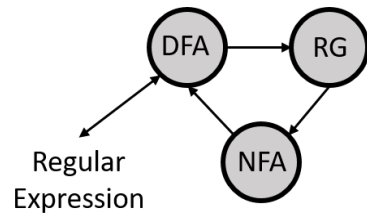
13.7 正規表示法

$L : \{b, ab, aab, aaab, \dots\} = \{a^k b \mid k \geq 0\} = a^*b$

$a^* : \lambda, a, aa, \dots$ // a^+ 為 $a \cdot a^*$

$a \cdot b$: 串接

$a+b$: 或，也可記 $a \vee b$

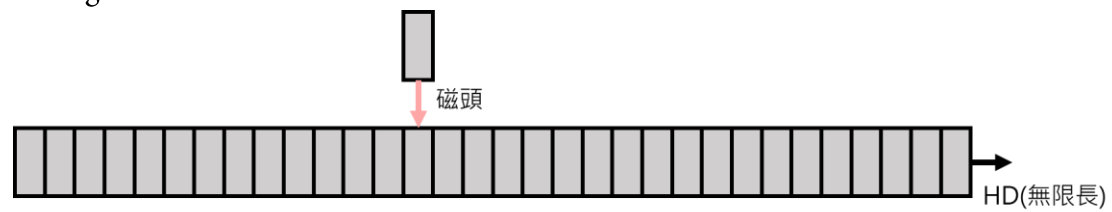


$_0101\dots = (0V1)(01)^*(01)$

例 : $L : \{a^k b^k \mid k \geq 0\}$ 無法用 Regular Expression 寫出 \Leftrightarrow 無法用 DFA, NFSA 表示

13.9 Turing Machine

Turing Machine



Turing Machine 之能力即為當今電腦之能力(第一代電腦)

Turing Machine 無法解 Halting Problem(某個程式無法決定另一程式在某 Input 下之 Output)

例：可 Compile 所有程式之 Compiler 不存在，因為那個 Compiler 由誰來 Compile？

第二代電腦：可能是量子電腦(目前只有理論，無實作出)