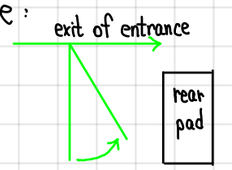
Finite Automata (有限自動機)

example:



② this is a one way door

f	r	
o	x	→ open
x	o	→ open
o	o	→ open
x	x	→ closed

two state : {open, closed}

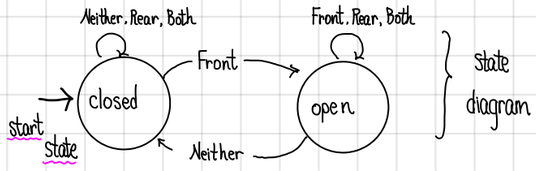
but what about the fromer state

same as

Door State	Neither	Front	Rear	Both
closed	closed	open	closed	closed
open	closed	open	open	open

state transition table

one-way exit — prevent injury



state diagram

⇒ What difference between Calculus

→ to judge but no ~~numbers~~ not main point

→ no input no output

What is input ⇒ a bund of parameters

⇒ Input = {N, F, B, R}

⇒ NFNFBRBF .....

Input → func. → Output (State)

{N, F, B, R}

• No Initial States → May have different Output

why

example : {FR}

Initial States

⇒ closed ⇒ closed

⇒ open ⇒ open

have 2×256 possibilities

Finite Automata

Numerical Definition:

5 tuple : $Q, \Sigma, \delta, q_0, F$

Q : a finite set called the states

Σ : a finite set called the alphabet

δ : $(Q \times \Sigma \to Q)$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept states / final states → symbol ◯

What if : States nums > Output nums

→ classify into Output nums groups

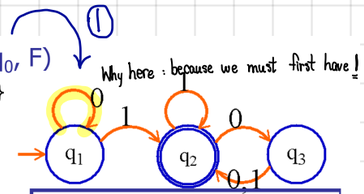
Example:

$M_1 = (Q, \Sigma, \delta, q_0, F)$

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- δ :

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

- q_1: the start state
- $F = \{q_2\}$



L(M) = A
M recognizes A or
M accepts A
the set of all strings that M accepts

$L(M_1)$ = ?

Important : able to do ① and ② {重點

How to turn to Human Language ?

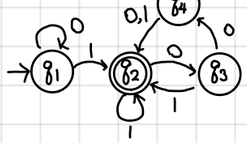
→ the last digit is "1"

→ the last 1 backward has even 0

Example 2:

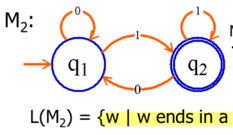
1000.....

num = 3·k $k \in \mathbb{N}$ (include 0)



Example

M_2:



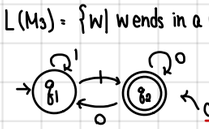
$M_2 = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$

δ:	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_2) = \{w \mid w \text{ ends in a } 1\}$

5

$L(M_3) = \{w \mid w \text{ ends in a } 0\}$

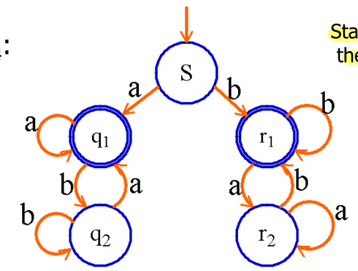


change too much ← better

② consider ϕ null string

because start q_1, end immediate on q_1

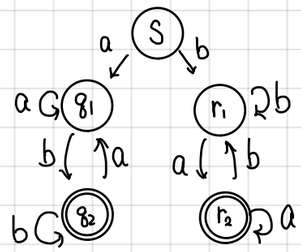
M_4:



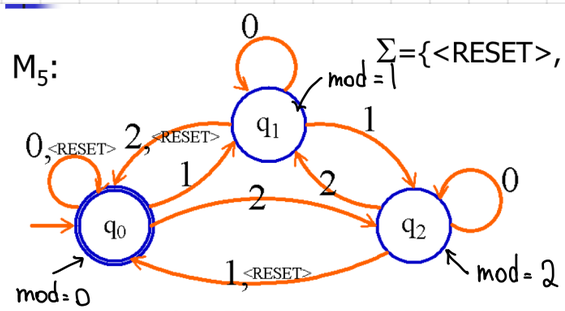
Starts and ends with the same symbol

7

→ 開頭和結尾不一樣



M_5:



$\Sigma = \{<RESET>, 0, 1, 2\}$

1 0 <RESET> 2 2 <RESET> 0 1 2

What is the language accepted by the above automata?

8

Observe : Reset → back to q_0 States num Important props

• Why we have 3 states → 3的餘數

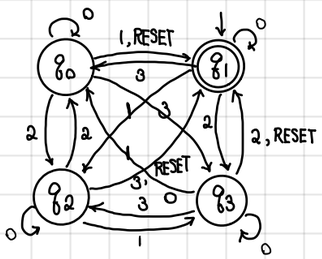
$L(M_5) = \{w \mid$ the sum of the symbols in w is 0 mod 3 excepts the <RESET> resets to 0$\}$

$L(M_5) = \{w \mid$ the sum of the symbols in w is 1 mod 3 excepts the <RESET> resets to 0$\}$

⇒ reset change to q_1

$L(M_5) = \{w \mid$ the sum of the symbols in w is 1 mod 4 excepts the <RESET> resets to 1$\}$

⇒ Need to add a state



Context of Computer Science :

→ a word is the concatenation of symbols

→ The used symbols are called alphabet

ex: words : "1", "1000", "543"

are formed out of the alphabet {0,1,2,3,4,5}

→ A language is a subset of all possible words

ex: define a language that caputers all MI6 elite agents

(means those start with double-0)

so words in language could be 007, 005 but not 01

Formal definition of computation

→ $M = (Q, \Sigma, \delta, q_0, F)$

→ w : a string over Σ, $w_i \in \Sigma$, $w = w_1, w_2, \dots w_n$

• M accepts w if a sequence of states $r_0, r_1, \dots, r_n$ exists in Q (n+1個 states)

& • $r_0 = q_0$

• $\delta(r_i + w_{i+1}) = r_{i+1}$, for $i = 0, 1, \dots, n-1$ ⇒表示 r_i 經過 transition function 後會到 r_{i+1} states

• $r_n \in F$

• We say that M recognizes language A if $A = \{w | M \text{ accepts } w\}$

• A language is called regular language if some finite automata recognizes it

• When designing a finite automata think:

→ What is important information?

→ state 的狀態會把 important information 記起來

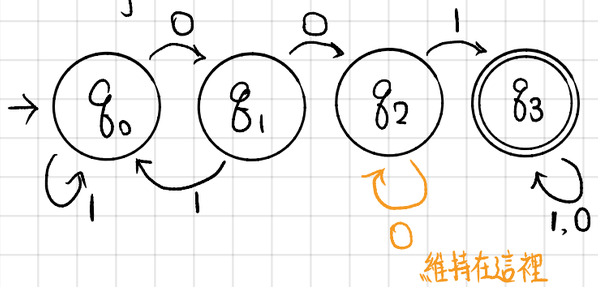
當 Design 一個判斷 even, odd 的 finite automata

→ 一般程式 加起來除2 → 這是一種 infinite 的 automata

↳ 加起來的東西是未知的必須用 Memory 來存

→ 一個一個讀是 finite 的 automata 的形式。

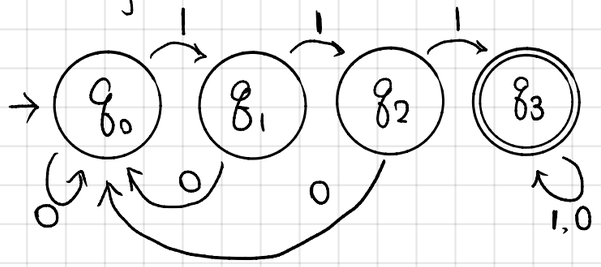
ex: Design a finite automata to recognize all strings that contains 001 as a substring



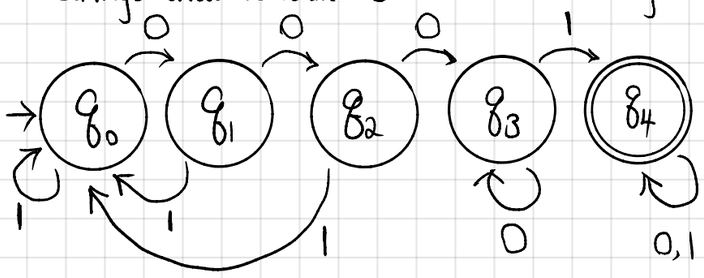
• 因為要吃 001 3個 sequence (n)

→ 所以要有 4個 states (n+1)

ex: Design a finite automata to recognize all strings that contains 111 as a substring



ex: Design a finite automata to recognize all strings that contains 0001 as a substring



Conclusion: 找 substring 是一種 regular language

↓ 由此去推論

用 operation 來推不同的 regular language

要知道怎麼寫

• Union : $A \cup B = \{x | x \in A \text{ or } x \in B\}$

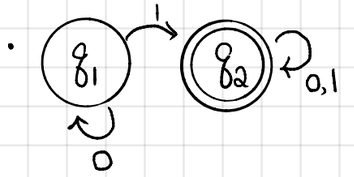
• Concatenate : $A \circ B = \{xy | x \in A \text{ and } y \in B\}$

• Star : $A^* = \{x_1 x_2 x_3 \cdots x_k | k \geq 0 \text{ and each } x_i \in A\}$

ex: $\Sigma = \{a, \dots, z\}$ $A = \{good, bad\}$ $B = \{boy, girl\}$

$A^* = \{\emptyset, goodgood, badgirl, badbad bad, \dots\}$.

ex: $A_2 = \{w | w \text{ has a } 1\}$



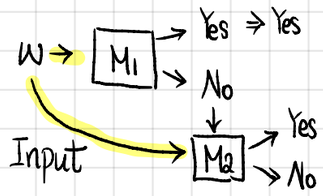
$A_1 \cup A_2 = \{w | w \text{ has odd number of } 1 \text{ or } w \text{ has at least of } 1\}$

因為是 Union ∴ ↳ w 必定有 1 則 A_2 成立

A_1 包含於(⊂) A_2, A_1 是 A_2 的子集

$A_1 \subset A_2$ ⇒ ∴ Design A_2 is alright

How Can we know $w \in A_1 \cup A_2$ without knowing A_1, A_2



← This isn't a finite automata ∵ Input 可能讀 2次

→ 為了能看到全部的 state，那就是了解到 A_1, A_2 的 state pair

⇒ $Q = \{(q_{even}, q_1), (q_{even}, q_2), (q_{odd}, q_1), (q_{odd}, q_2)\}$

$\Sigma = \{0, 1\}$

δ:

	0	1
(q_{even}, q_1)	(q_{even}, q_1)	(q_{odd}, q_2)
(q_{even}, q_2)	(q_{even}, q_2)	(q_{odd}, q_2)
(q_{odd}, q_1)	(q_{odd}, q_1)	(q_{even}, q_2)
(q_{odd}, q_2)	(q_{odd}, q_2)	(q_{even}, q_2)

• (q_{even}, q_1) : start state

• (q_{odd}, q_1) (q_{odd}, q_2) (q_{even}, q_2) : accept state

(1) 造一機器.

(2) Intersection → 寫一機器.

Conclusion:

- Pf: Let
  M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
  Goal: Construct $M=(Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$
  - $Q=\{(r_1,r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
  - Σ is the same in M_1, M_2
  - For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let $\delta((r_1,r_2), a) = (\delta_1(r_1,a), \delta_2(r_2,a))$
  - $q_0=(q_1, q_2)$, (q_0 is a new state $\notin Q_1 \cup Q_2$)
  - $F=\{(r_1, r_2) | r_1 \in F_1$ **or** $r_2 \in F_2\}$

5 tuple the new finite automata

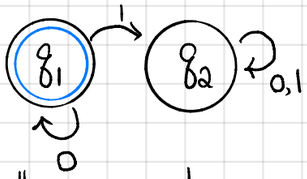
20

- How about $A_1 \cap A_2$? & $A_1 \oplus A_2$?

→ $F=\{(r_1,r_2) | r_1 \in F_1 \text{ and } r_2 \in F_2\}$ → $F=\{(r_1,r_2) | r_1 \in F_1 \cap r_2 \notin F_2 \text{ or } r_1 \notin F_1 \cap r_2 \in F_2\}$

How about A_1, $\overline{A_1}$

→ Change Finale state to another one ⇒ 以 A_2 舉例



⇒ Theorem:

The class of regular languages is closed under the union operation

& " " " Intersection "

& " " " Exclusion "

Why Intersection maybe

Operation Concatenation:

Use the way Union operate → ① $s \in A_1 \cap w \in A_2$ ② $w \in A_1 \cap w_2 \cdots w_n \in A_2$

⇒ We may have infinite sequences

→ infinite states ⇎ finite automata

⇒ the limitation of the finite automata

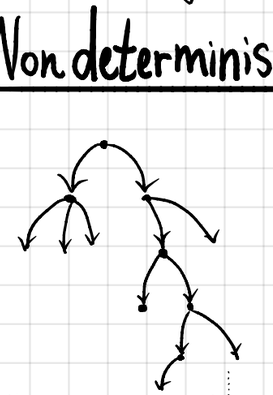
Σ	W
w_1	$w_2 \ldots w_n$
$w_1 w_2$	$w_3 \ldots \ldots w_n$

→ We hope as we operate W_1, W_2

→ We let $W_3 \ldots W_n$ may auto operate.

So Far we manage Determinism Computation:

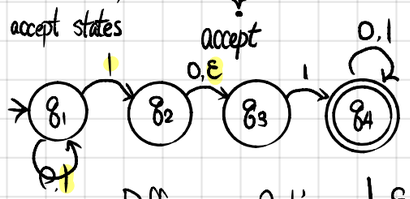
↳ **Nondeterminism**:



左邊出去
右邊進來

But this is still automata
→ Will be discussed

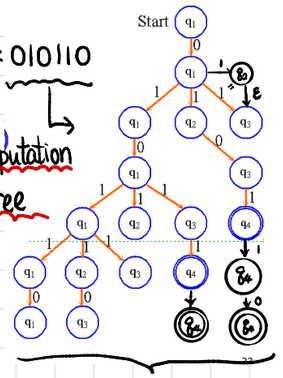
If there is a way to walk to accept states



Difference: 2 1's and ε, q_3 has no 0

meaning we may jump to another states without accepting any sequence

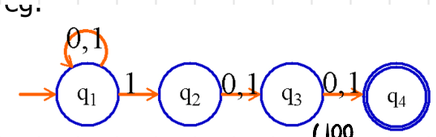
Input: 010110



會考 Computation tree

Reject 沒有一條是 Accept 的

- e.g.



看不出規律 → 先舉例 100, 010100, 0101 11, 0101 10

→ q_1 ⇒ 不確定 Input 到底結束了沒

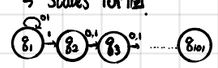
→ 等待最末尾的到來

→ 倒數第3位為1

↖ state 輔助

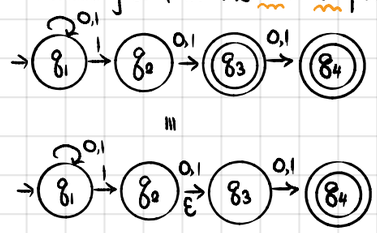
若倒數100呢

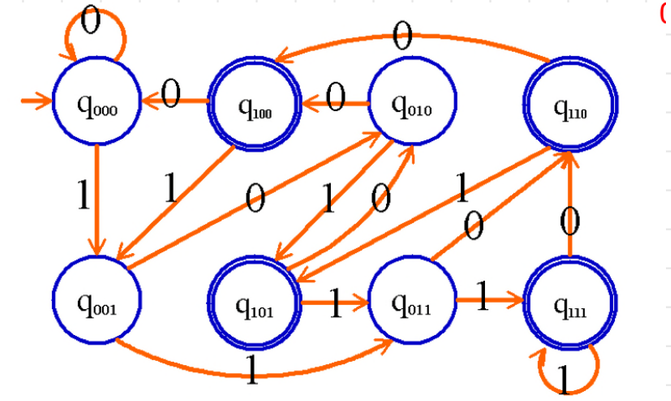
→ states 101個



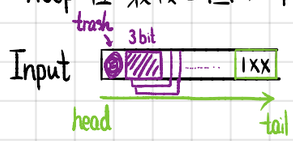
Non-deterministed automata ✱ not so simple deterministed

② containing a 1 in the 3 or 2 position

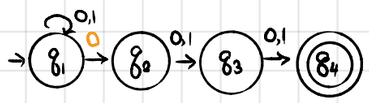




↖ Keep 住 最後3個 → {1,0}, $2^3 = 8$



- containing a 0 in the 3 position



↳ 在 deterministed automata

→ 從 111 開始

→ final state 改成有0的

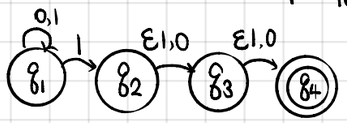
- NFA, DFA ⇝ 都要會畫

Nondeterministed Final Automata → 有便利性 → 節省資源

Deterministed Final Automata ↳ 較易設計、改動

Discuss

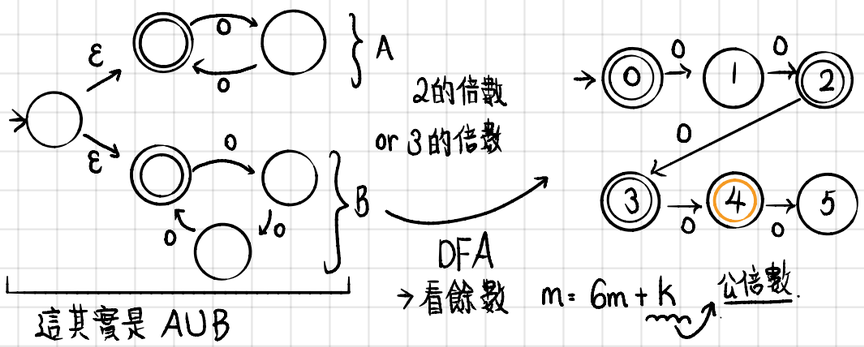
ε-Move add to have {~1, ~10, ~100



DFA

⇒ 除了 q_{000} 都可是 final accept state

→ 3個1的 NFA



→看餘數 $m = 6m + k$ 公倍數.

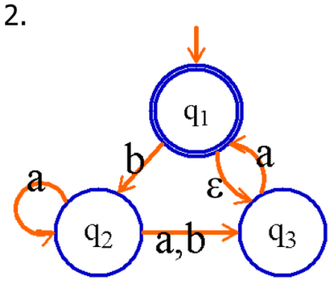
這其實是 A∪B

- accept all strings of 0^k $k \in$ 2,3的倍數

↳ NFA not 比較難實現

accept:

2.



- ε ✓
- a ✗
- baba ✓
- baa ✓
- b ✗
- bb ✗
- babba ✗

DFA ??

⇒ Unable to determine What this DFA is doing.

- $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$,
  $\mathcal{P}(Q)$: the collection of all subset of Q
- A nondeterministic finite automaton is a 5-tuple (Q, Σ, δ, q_0, F), where
  - Q : a finite set of states
  - Σ : a finite set of alphabet
  - δ : $Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ transition function
  - $q_0 \in Q$: start state
  - $F \subseteq Q$: the set of accept states

a representation of set

$\delta(q, a) = \{q_{11}, q_{12}, \dots\}$

q的 power set

cmp DFS

Q
Σ
δ
q_0
F

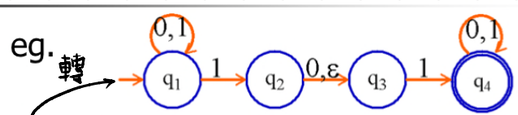
$Q \times \Sigma \to Q$

30

δ 全部都可以用 set 表示 $\{\phi, q_1, q_2, \dots\}$ 沒有地方去

- Power is a set containing all the subsets of the given set.

- eg.



- Q = {q_1, q_2, q_3, q_4}
- Σ = {0, 1}
- δ :

	0	1	ε
q_1	{q_1}	{q_1, q_2}	ϕ
q_2	{q_3}	ϕ	{q_3}
q_3	ϕ	{q_4}	ϕ
q_4	{q_4}	{q_4}	ϕ

No where to go

- q_1 : start state
- q_4 : accept state

- $N = (Q, \Sigma, \delta, q_0, F) \sim$ NFA

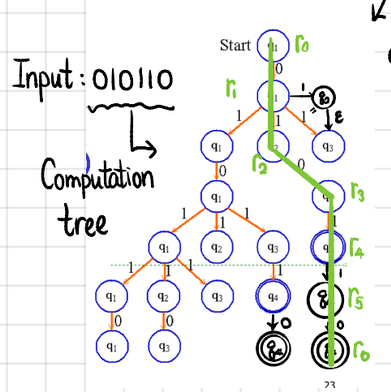
w : string over Σ, $w = y_1 y_2 y_3 \dots y_m, y_i \in \Sigma_\varepsilon$

$r_0, r_1, r_2, \dots, r_m \in Q$

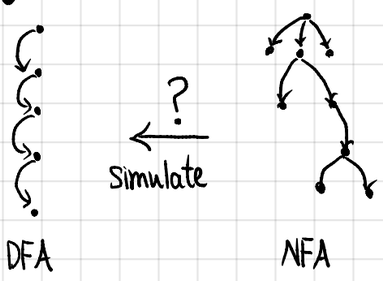
- N accepts w if exists $r_0, r_1, r_2, \dots, r_m$ such that
  - $r_0 = q_0$ ~ 起始
  - $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \dots, m-1$

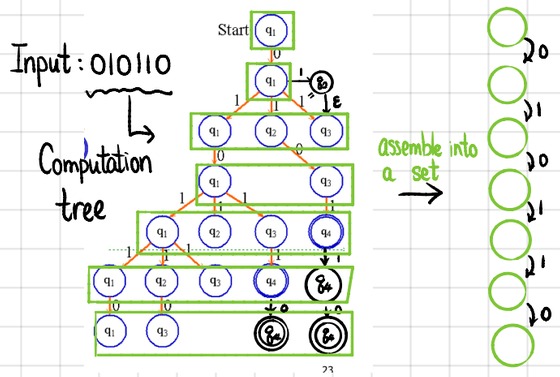
  $r_i \to q_i$ transit to q_i
  - $r_m \in F$

↳ 對應 the definition



Equivalence of NFA and DFA





Theorem : Every nondeterministic finite automaton has an equivalent deterministic finite automaton

Pf: Let N=(Q, Σ, δ, q_0, F) be the NFA recognizing A.

Goal: Construct a DFA recognizing A

1. First we don't consider ε.
   Construct M= (Q', Σ, δ', q_0', F')
   1. Q'= $\mathcal{P}(Q)$
   2. For R∈Q' and a∈Σ
      let $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a), r \in R\}$
      $= \bigcup_{r \in R} \delta(r, a)$ → Union
   3. q_0'= {q_0}
   4. F' = {R∈Q' | R contains an accept state of N}

33

E(R)= R∪R∪R

Consider ε :

For any R∈M, define
E(R)={q | q can be reached from R by traveling along 0 or more ε}

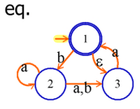
Replace δ(r,a) by E(δ(r,a))

Thus,
δ'(R, a)={q∈Q : q∈E(δ(r,a)) for some r∈R}

Change q_0' to be E({q_0})

- Corollary: A language is regular if and only if some NFA recognizes it

eq.



D's state :
{ϕ, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}

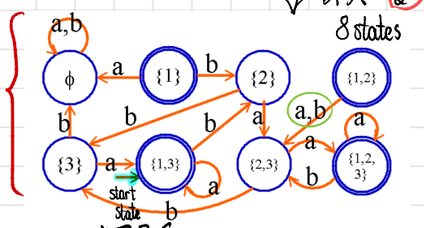
N_4 = ({1,2,3}, {a,b}, δ, 1, {1})

accept state

35

取 or 不取 {1,2,3}

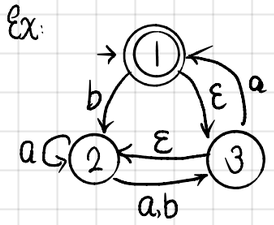
↓ DFA 2^3=8 states



主因是 ε

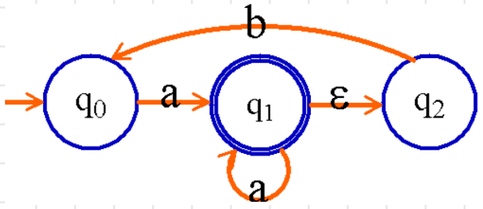
	a	b
ϕ	ϕ	ϕ
{1}	ϕ	{2}
{2}	{2,3}	{3}
{3}	{1,3}	ϕ
{1,2}	{2,3}	{2,3}
{1,3}	{1,2,3}	{2}
{1,2,3}	{1,2,3}	{2,3}
{2,3}	{1,2,3}	{3}

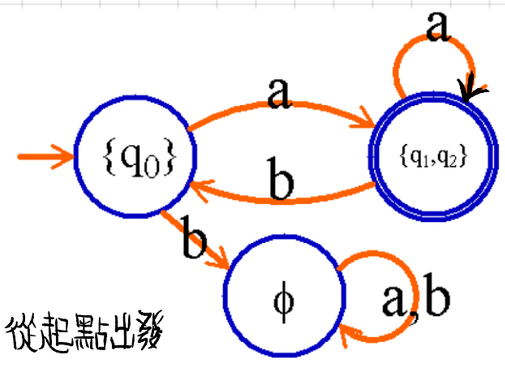
Ex:



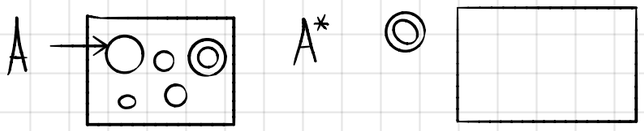
	a	b
ϕ	ϕ	ϕ
{1}	ϕ	2
{2}	{2,3}	3
{3}	{1,2,3}	ϕ
{1,2}	{2,3}	{2,3}
{1,3}	{1,2,3}	{2}
{1,2,3}	{1,2,3}	{2,3}
{2,3}	{1,2,3}	{3}

- 觀察
  → 哪一個是多餘
  → 起點到不了的
  → 從起點出發,有走到的才建立





A* operation :

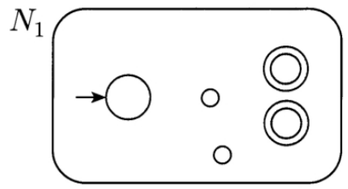


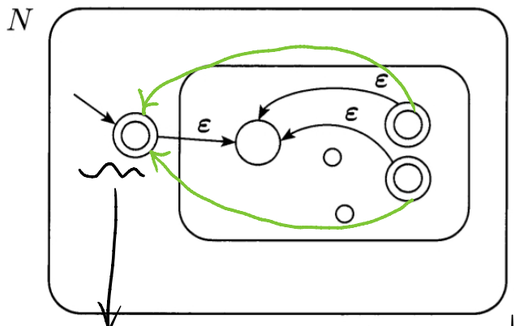
$A = \{a, b\}$

$A^* = \{\varepsilon, a, b, \dots\}$

① Simple Regular language

② Generated Regular language





we shouldn't touch the machine

reason: add null ($\emptyset$)

Regular Expression

we want to know the language accept what kind of state

→ this kind of Expression is Regular Expression

$(0 \cup 1)^* = \{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, \dots\}$

→ 所有字串的集合

Definition :

R is a regular expression if R is

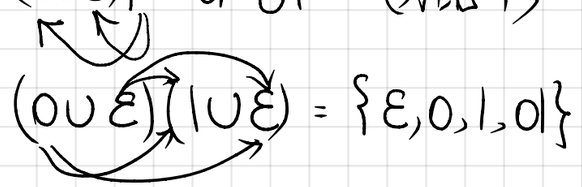
- $a \in \Sigma$
- $\varepsilon = \emptyset^*$
- $\emptyset$
- $(R_1 \cup R_2)$, R_1 and R_2 are regular
- $(R_1 \circ R_2)$, R_1 and R_2 are regular
- (R_1^*), R_1 is regular expression

ex) $(a \cup \varepsilon)^*$

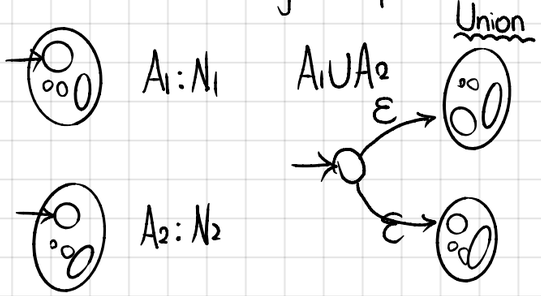
→ $(\emptyset)^* \circ (a \cup \varepsilon)^* \circ b^*$ a regular expression

ex $\Sigma = \{0,1\}$

- $0^*10^* = \{w: \text{whas exactly a single 1}\}$
- 連續2個1 : 0^*110^*
- $\Sigma^*1\Sigma^*$: 至少一個1
- $\Sigma^*001\Sigma^*$: 至少一個001
- $(\Sigma\Sigma)^*$: 長度為偶數
- $01 \cup 10$ $\{01, 10\}$ 由01, 10構成字串
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$ 開頭和結尾相同
- $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$ (分配率)



- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
- $1^*\emptyset = \emptyset$ 不存在
- $\emptyset^* = \{\varepsilon\}$
- $1 \to \varepsilon$ $\{\sim\} \circ \{\varepsilon\} = \{\sim\}$ ($\varepsilon \equiv 1$)

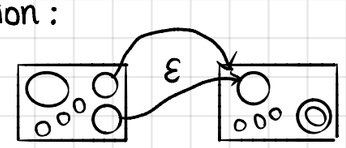


How about $A_1 \cap A_2$

→ DeMorgan Rule $\overline{\neg(A_1 \cap A_2)} \to \overline{\neg A_1 \cup \neg A_2}$

- Concatenation :

$A_1 \circ A_2$:



原本的final state

R : regular expression

$R \cup \phi = R$

$R \circ \varepsilon = R$

$R \cup \varepsilon$? R

→ R 不含 ε 則 不正確

$R \circ \phi$?= R

→ 加 0 則正確.

$R \circ \phi = \phi$

#Theorem :

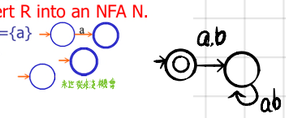
A language is regular if and only if some regular expression describes it

- If a language is described by a regular expression then it is regular. (←)

Let R be a regular expression describing some language A

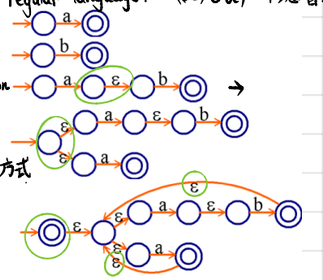
Goal: Convert R into an NFA N.

- R=a ∈ Σ, L(R)={a}
- R=ε, L(R)={ε}
- R=ϕ, L(R)=ϕ
- $R = R_1 \cup R_2$
- $R = R_1 \circ R_2$
- $R = R_1^*$



eg. (ab ∪ a)* regular language. →(ab)∪a)* 不應省略

- a
- b
- ab concatenation
- ab ∪ a union (NFA)的方式
- (ab ∪ a)*



- R: Regular expression

L(R) : the language it describes

- $1^*(01^+)^*$: every 0 follows at least one 1

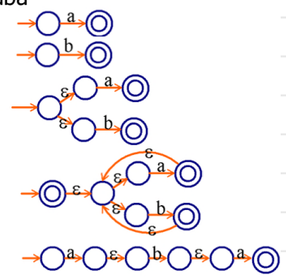
a^+ means $\{a^n \mid n \geq 1\}$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$.
2. $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$.
3. $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring}\}$.
4. $1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.
5. $(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}$.[5]
6. $(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of three}\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$.
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$.
   The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^*.
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.
    Concatenating the empty set to any set yields the empty set.
12. $\emptyset^* = \{\varepsilon\}$.
    The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.

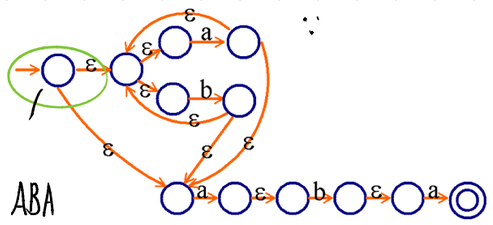
考試給一張轉-NFA

eg. (a ∪ b)*aba

- a
- b
- a ∪ b
- (a ∪ b)*
- aba



→ 不做優化



∵ $(A \cup B)^* \neq ABA$

有一DFA 有 10 state

↓轉換成

SNFA (10 states) → 降 states ……

special ↓

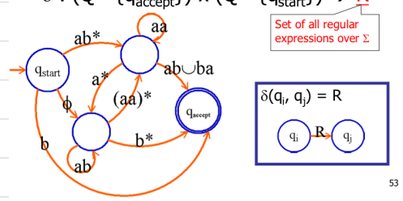
GNFA

Generalized

GNFA

- (Q, Σ, δ, q_{start}, q_{accept})
  δ : (Q - {q_{accept}}) x (Q - {q_{start}}) → R

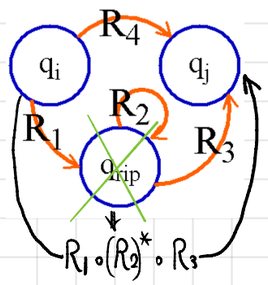
Set of all regular expressions over Σ



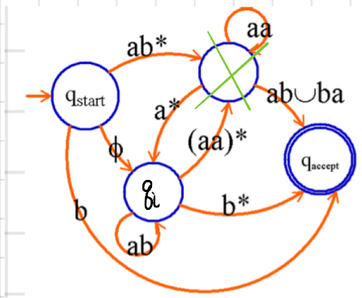
53

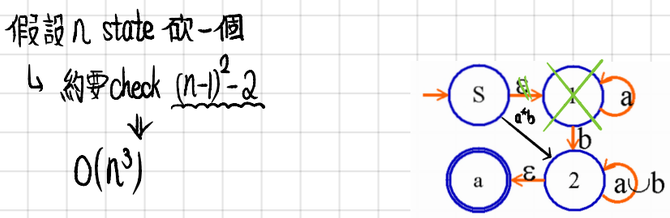
- 問題：如何讓有 2個 accept state 變成一個.

- 先取任兩個 state



$(R_1(R_2)^* \circ R_3) \cup R_4$





→剛好相同

假設 n state 砍一個

↳ 約要check $(n-1)^2-2$

↓

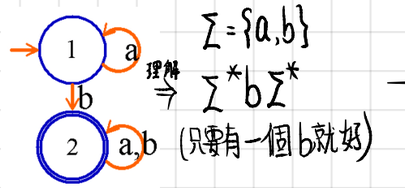
$O(n^3)$

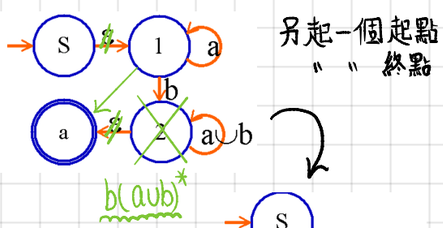
ϕ v.s ε 的差別

ε → 無條件進位 (跳方法)

ϕ → 沒有東西

(example)





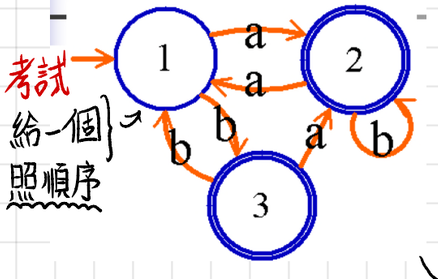
A GNFA accepts a string w in Σ^* if $w = w_1 w_2 \ldots w_k$, where each w_i is in Σ^* and a sequence of states $q_0 q_1 q_2 \ldots q_k$ exists s.t.

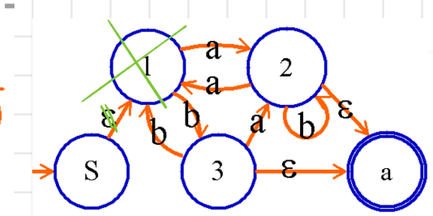
- $q_0 = q_{start}$ is the start state
- $q_k = q_{accept}$ is the accept state
- for each i, we have $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$

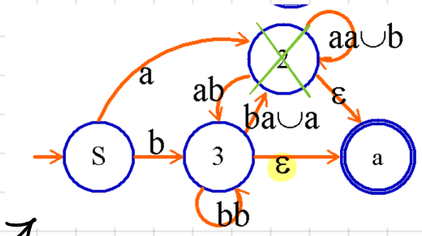
DFA M ----------→ GNFA G

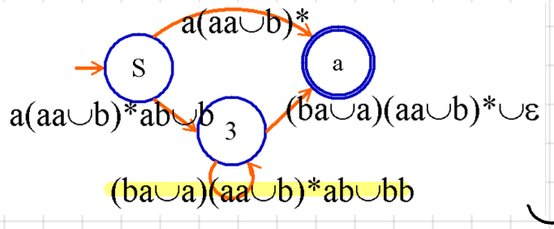
adding { start state / accept state / transition arrows

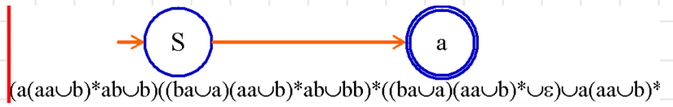
Convert (G) : takes a GNFA and return an equivalent regular expression











→期中考考到這

◎有沒有DFA無法辨視

↓

state為有限

有限種資訊

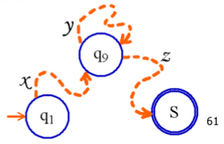
- Is {0^n1^n: n≥0} regular?
  How can we prove a language non-regular?

a form of infinite

↓

DFA無法處理

- Theorem: (Pumping Lemma)
  If A is a regular language, then there is a number $\mathcal{P}$ (the pumping length) where, if s is any string in A of length≥$\mathcal{P}$, then s may be divided into 3 pieces, $s=xyz$, 因為夠長 satisfying the following conditions:
  - for each $i \geq 0$, $xy^iz \in A$
  - $|y| > 0$
  - $|xy| \leq \mathcal{P}$



→ strictly proof ?

$\mathcal{P}$: pumping length

→ pumping lemma

↳描述1個DFA