204753

Theory of computation

Lecture 3: Nondeterminism

- Review
- Nondeterminism
- Equivalence of NFAs and DFAs
- Closure under the regular operations

Regular operations

คราวก่อนได้นิยาม 3 regular operations:

Let A and B be languages. We define the regular operations **union**, **concatenation**, and **star** as follows.

- Union: $A \cup B = \{x | x \in A \text{ or } x \in B\}$
- Concatenation: $A \circ B = \{xy | x \in A \text{ and } y \in B\}$
- Star: $A^* = \{x_1 x_2 \ldots x_k | k \geq 0 \text{ and each } x_i \in A\}$

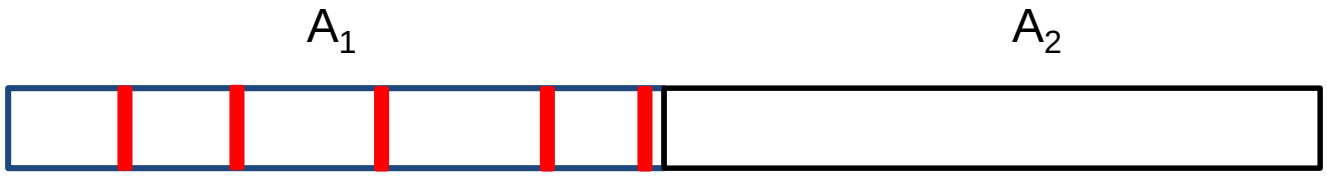
คราวที่แล้วเราได้แสดงว่า regular language นั้นมีคุณสมบัติปิดภายใต้ union

โดยเราได้ทำการจำลอง 2 finite automata ด้วย 1 finite automaton

แต่วิธีการข้างต้นไม่สามารถนำมาใช้บอกว่า set ของ regular language นั้นมีคุณสมบัติปิดภายใต้ concatenation

สังเกตว่า union มันทำไปพร้อมกัน ทำให้ simulate พร้อมกัน

สำหรับ string $w \in A_1 \circ A_2$ พบว่ามีคู่ของ x และ y ที่ $w = xy$ และ $x \in A_1$ และ $y \in A_2$



ในการสร้าง finite automaton M สำหรับ $A_1 \circ A_2$ จาก M_1 และ M_2 ที่ recognize A_1 และ A_2 นั้นเราต้องการจำลอง M_1 ที่หยุดตอนท้ายของ x แล้วเริ่มต้นทำงาน M_2 ต่อจากนั้นเลย

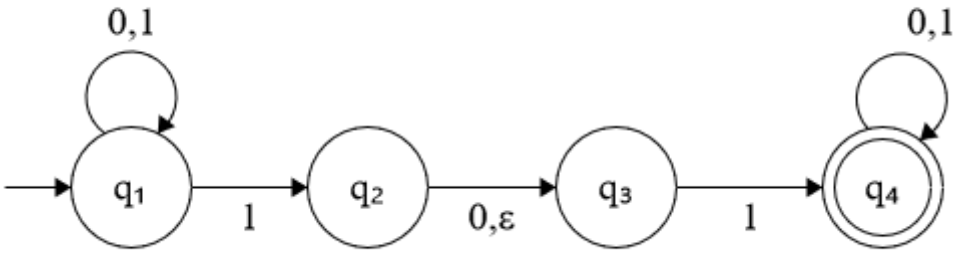
มันยากตรงที่จะบอกว่าจุดไหนที่ x หยุดแล้วควรไปต่อ เพราะว่าระหว่างทางมันอาจจะผ่าน accept state เยอะแยะมากมาย

สมมติว่า machine ของเราสามารถเดาได้ว่า x จบตรงไหน

ดังนั้นมันจะจำลอง M_1 บน input string ที่จบในส่วนของ x ได้

จากนั้นจะกระโดดไปยัง start state ใน M_2 หลังจากจบ x และ accept string w = xy เมื่อ machine หยุดที่ accept point บางอันของ M_2

Nondeterministic Finite Automaton N_1



ข้อแตกต่าง

- ลูกศรออกจาก state มี symbol ซ้ำ (เช่นมี 1 ออกสองอันใน q_1)
- symbol หาย (เช่น q_2 ถ้ารับ 1 มาไปไหนไม่รู้)
- มี empty string ϵ

ถ้าถึงจุดต้องเลือก ไม่รู้จะเดาทางไหน เช่นไป q_1 หรือ q_2 ดี ทำอย่างไรดี

(ลองดู simulation)

Deterministic and Nondeterministic Finite Automata

ก่อนหน้านี้เราพิจารณา finite automata ที่ state ถัดไปนั้นถูกตัดสินจาก input alphabet และ current state

การคำนวณซึ่งแต่ละขั้นถูกกำหนดอย่างแน่นอนจะถูกเรียกว่า **deterministic** computation

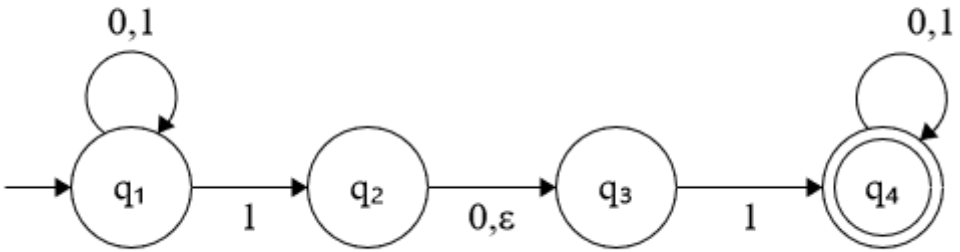
อีกด้านหนึ่งคือ ใน **nondeterministic** computation นั้น แต่ละขั้นอาจจะมีหลายทางเลือก

ดังนั้นเราจะเรียกให้ต่างกันไปเลยคือ deterministic finite automata(**DFA**) และ nondeterministic finite automata(**NFA**)

ข้อสังเกต

Nondeterminism is a generalization of determinism,

so every deterministic finite automaton is automatically a nondeterministic finite automaton



แต่ละจุดที่มีทางเลือกไปยังขั้นต่อไปหลายทาง machine จะ **แบ่ง**ตัวมันเองออกเป็นหลายๆ copy แล้วแยกกันไป**ทุก**ทางที่เป็นไปได้พร้อมๆกัน

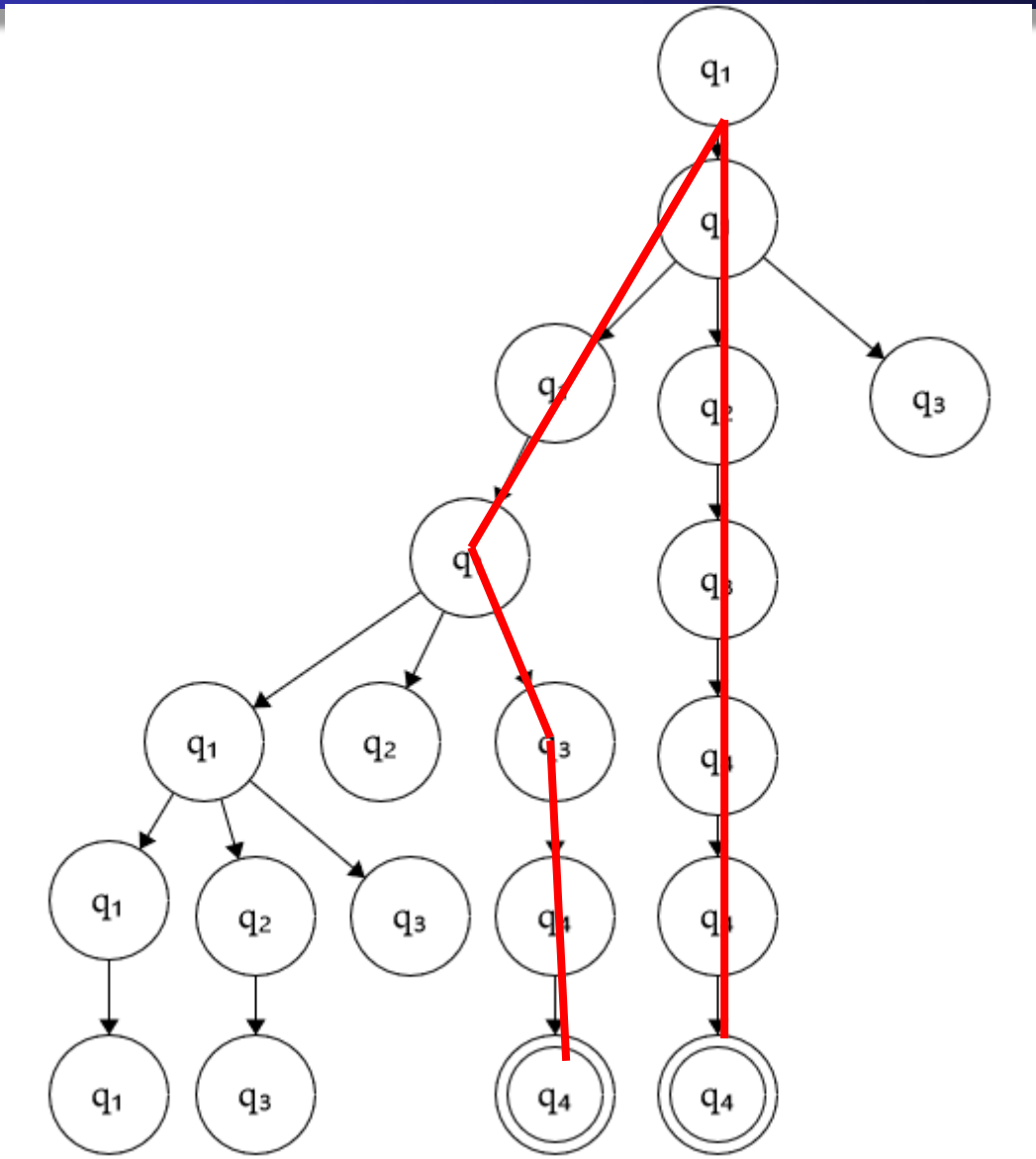
- ถ้ามีหลายทางเลือก ให้แบ่งตัวไปทำงาน

- ตัวที่แบ่งจะทำงานต่อไปเรื่อยๆ และจะตายถ้าไม่สามารถเดินต่อไปตาม input ที่ได้รับได้

- จะ accept string เมื่อไร

เมื่อ input หมดแล้วถ้ามีตัวที่แบ่งสักตัวหนึ่ง(any) ที่อยู่ใน accept state เราจะ **accept** input นั้น

N_1 เมื่อรับ input 010110

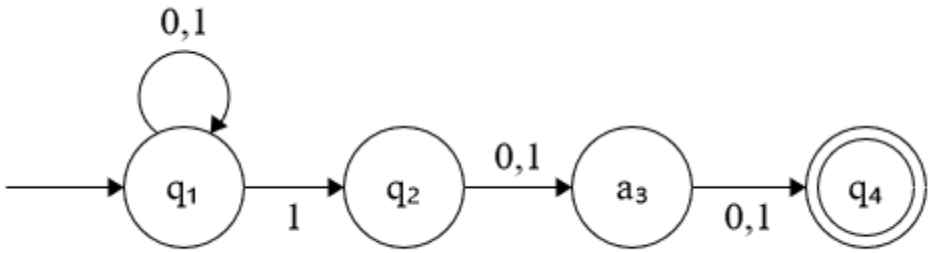


แบบนี้ computer จริงๆ ทำได้ไหม

ไม่น่าจะทำได้ แต่ถ้าเรายอมให้เครื่องทำได้ มันจะเก่งขึ้นไหม
ตัดสินปัญหาที่ยากขึ้นได้ไหม

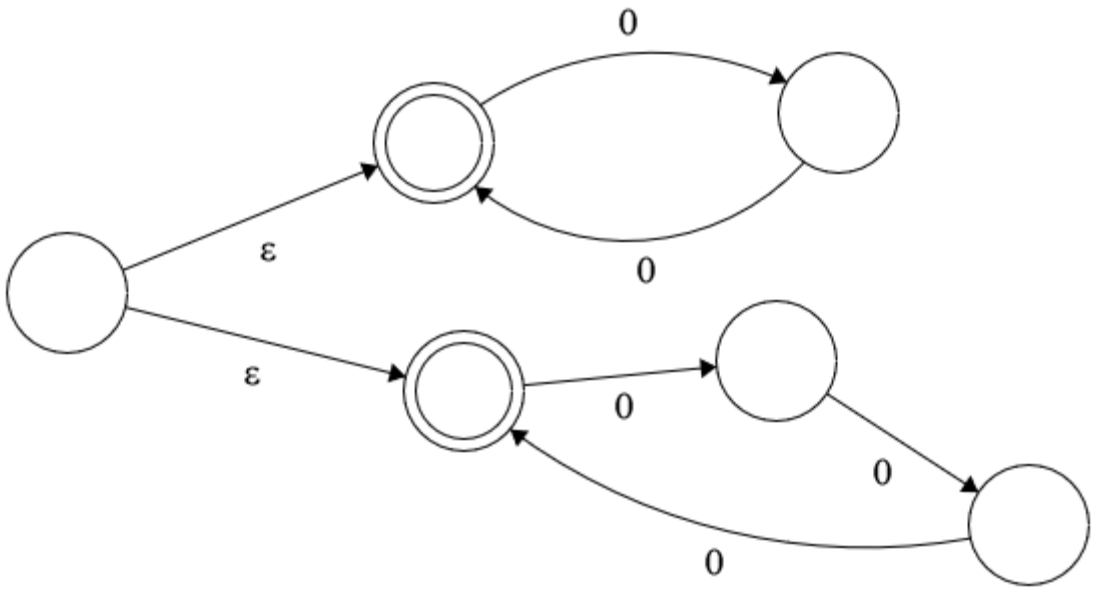
N_2

String ที่ N_2 accept คืออะไร



N_3

String ที่ N_3 accept คืออะไร ให้ {0} เป็น alphabet ของ N_3



Formal definition of NFAs:

หลักๆ ที่ต่างจาก DFA

Transition function δ:

รับ current state และ symbol ใน $\Sigma \cup \{\epsilon\}$

output เป็น subset ของ states Q

กำหนดให้ $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

แทน P(Q) ด้วย **power set** ของเซต Q

เราจะได้ว่า state transition δ สำหรับ NFA เป็นฟังก์ชันจาก

$$Q \times \Sigma_\epsilon \rightarrow P(Q)$$

Definition

A **nondeterministic finite automaton** is a 5-tuple

$(Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states

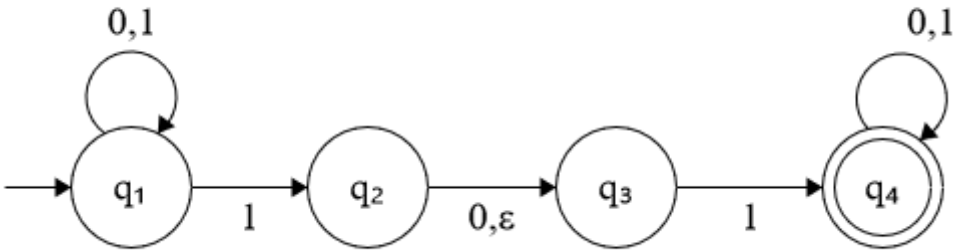
Σ is a finite alphabet

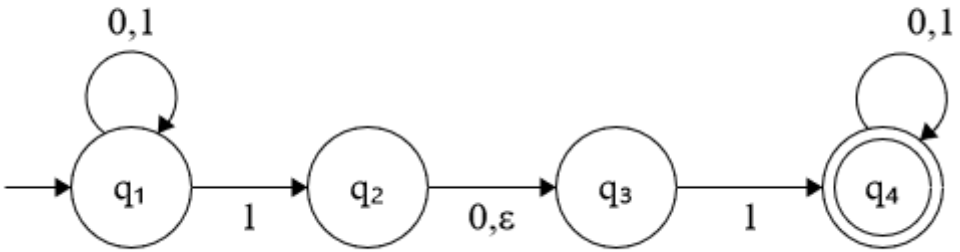
$\delta: Q \times \Sigma_\epsilon \rightarrow P(Q)$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept states

ให้เขียน formal definition ของ N_1





$N_1\ is\ (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_1, q_2, q_3, q_4\}$

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

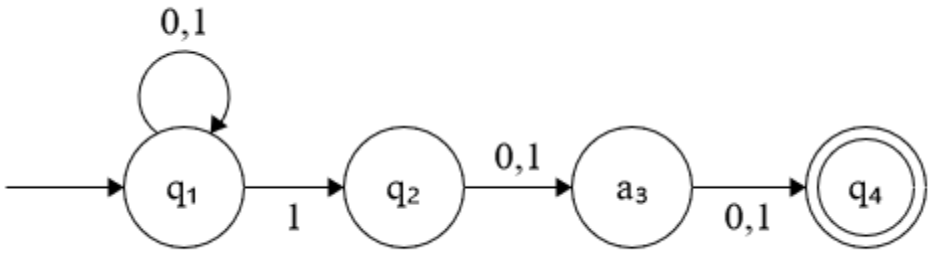
δ:

	0	1	ϵ
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

q_1 is the start state

$F = \{q_4\}$

ลองเขียน formal definition ของ N_2



Formal definition of computation of NFAs

ให้ $\mathrm{N} = (Q, \Sigma, \delta, q_0, F)$ เป็น NFA และให้ w เป็น string บน alphabet Σ

เราจะบอกว่า **N accept w** ถ้าเราสามารถเขียน $w = w_1 w_2 \ldots w_n$ ที่แต่ละ w_i เป็นสมาชิกของเซตอักขระ Σ_ϵ และมีลำดับของ state $r_0 r_1 \ldots r_n$ ใน Q ที่เกิดขึ้นมีเงื่อนไข 3 เงื่อนไขดังนี้

1. $\mathrm{r}_0 = q_0$
2. $r_{i+1} \in \delta(r_i, w_{i+1})$, for i = 0, 1, …, n-1 ← สังเกตว่านี่เป็นเซต
3. $r_n \in F$

NFA มีพลังมากกว่า DFA?

ด้วยพลังของ nondeterminism (ความไม่แน่นอน) ทำให้ดูเหมือนว่า NFAs มีพลังมากกว่า เหมือนว่ามันเดาได้

แต่ความเป็นจริงแล้ว DFAs และ NFAs นั้น recognize ภาษา class เดียวกัน

เราจะบอกว่า 2 machine นั้น equivalent กันถ้าพวกมัน recognize ภาษาเดียวกัน

ในการพิสูจน์ว่า equivalence

พิสูจน์ 2 ทาง

ทางแรก กำหนด DFA มาให้แล้วสร้าง NFA ที่ recognize ภาษาเดียวกัน ทางนี้ง่าย DFA เป็น NFA อยู่แล้ว (ทำไม)

ทางที่สอง กำหนด NFA มาให้แล้วสร้าง DFA ที่ recognize ภาษาเดียวกัน ทางนี้ไม่ง่าย

Theorem

Every nondeterministic finite automaton has an equivalent deterministic finite automaton

เราจะพิสูจน์อย่างไร

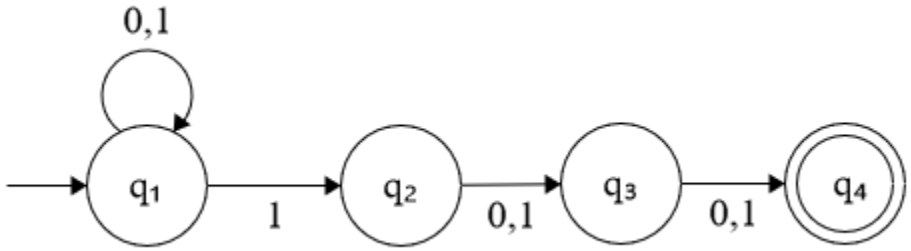
ทำตัวเป็น automaton

เมื่อกำหนดให้ NFA N มาให้คิดว่า DFA M เป็นคนใช้งาน N

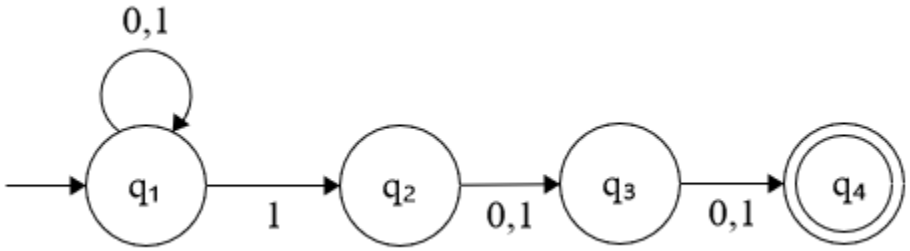
แล้วดูว่า M ต้องจำอะไรบ้างถึงจะใช้งาน N ได้

ตัวอย่าง

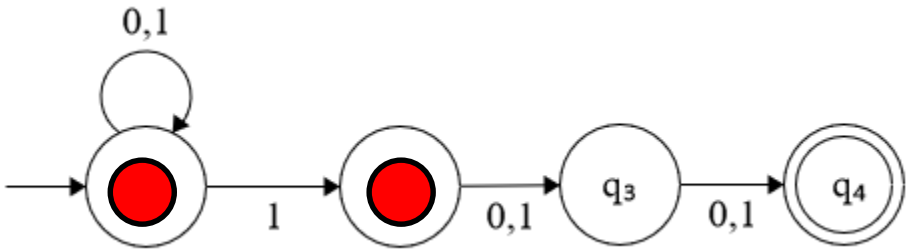
เพื่อความง่าย เราจะสนใจ



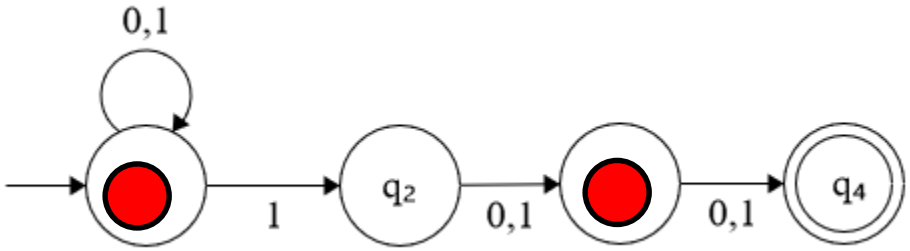
NFA นี้ รับอะไร



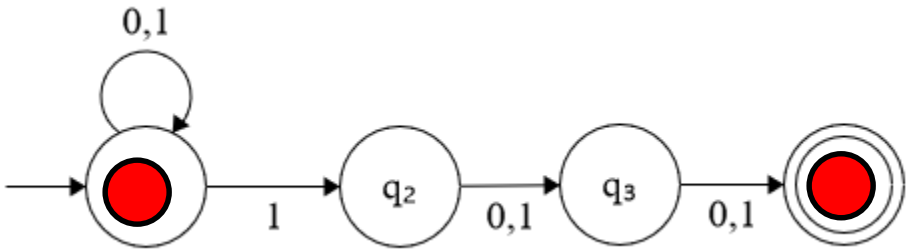
รับ 1 มาเกิดอะไรขึ้น



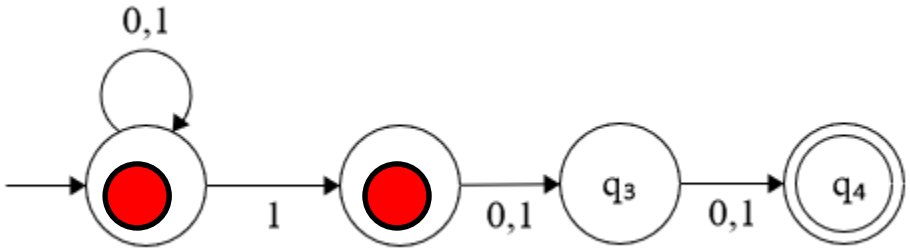
รับ 0



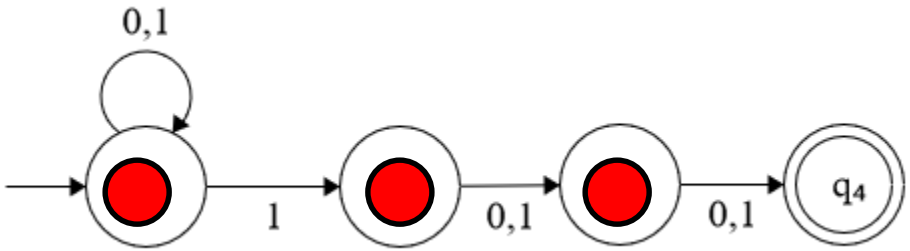
รับ 0



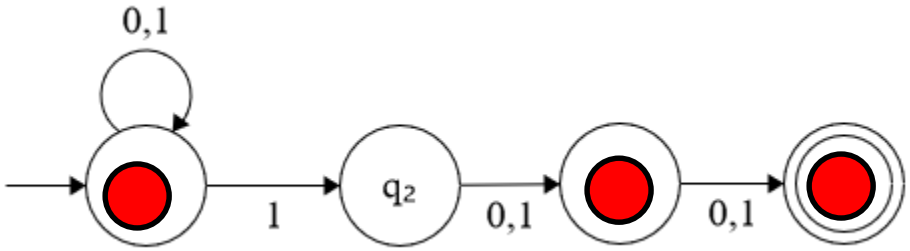
รับ 1



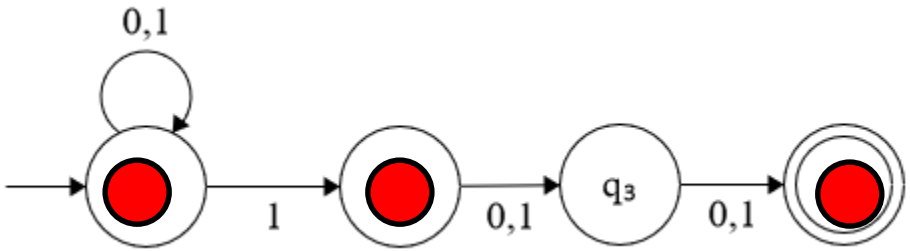
รับ 1



รับ 0



รับ 1



ต้องจำอะไรบ้าง

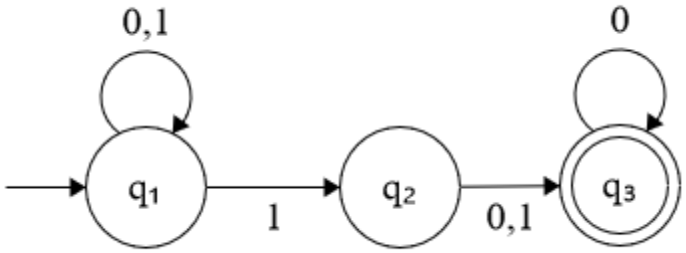
ต้องจำว่าแต่ละตัวอยู่ตรงไหนบ้าง

ทีนี้เรามี state กี่อัน

4

ดังนั้นเซตของตำแหน่งที่จะอยู่ได้มีกี่แบบ

มันก็ต้องเป็นสับเซตของ state ที่เรามี นั่นคือ 16 แบบ

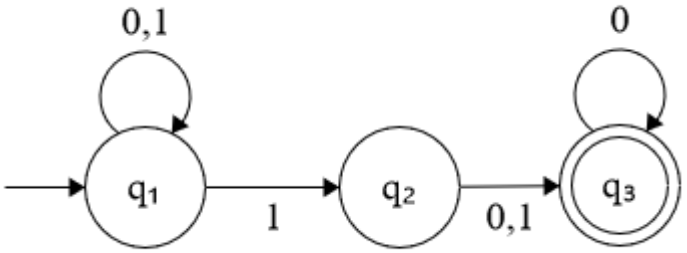


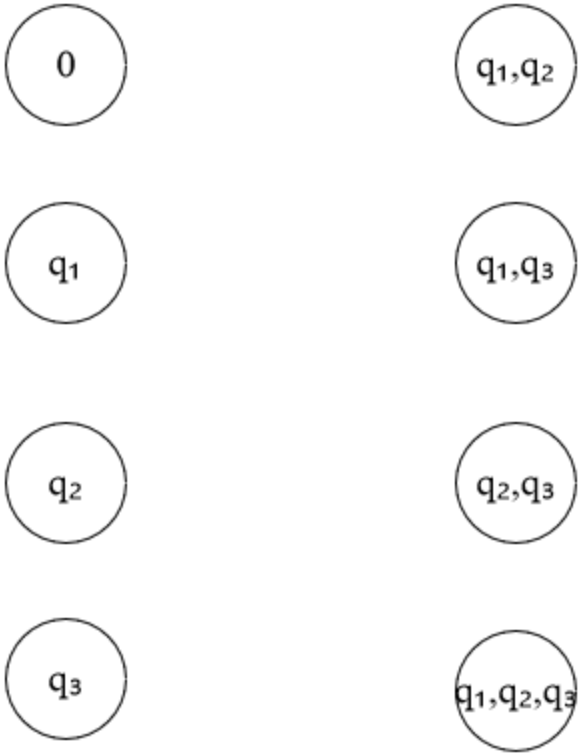
เราจะต้องจำว่า input แต่ละอันสถานะก่อนอยู่ที่ไหนบ้าง

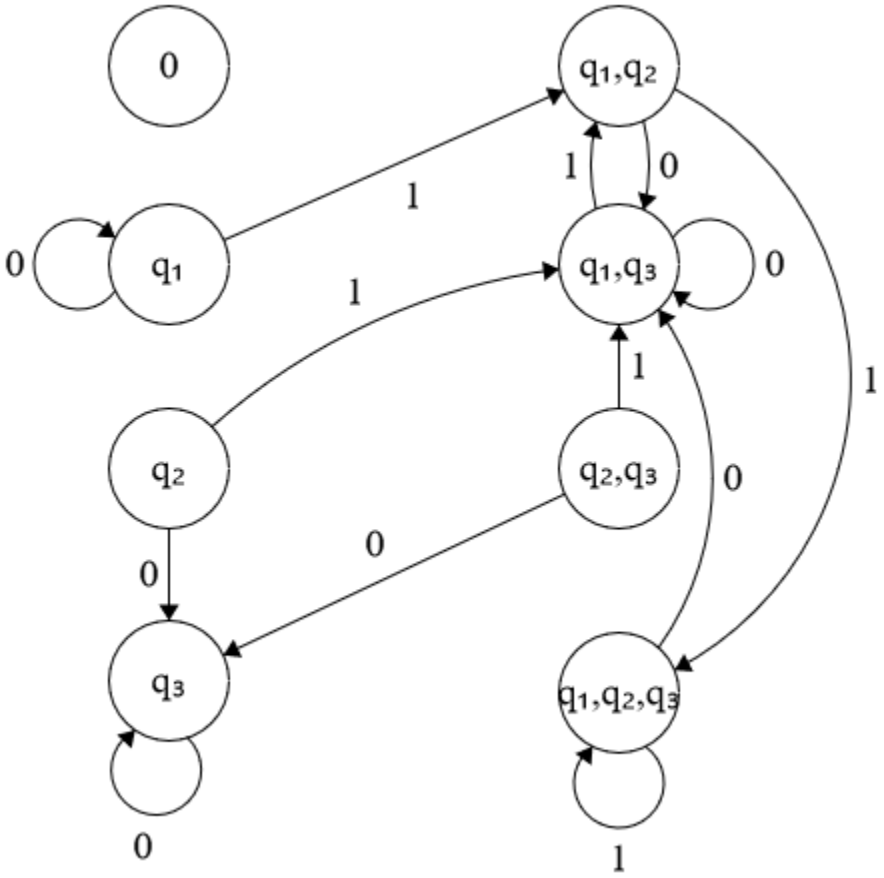
นอกจากนี้มันอาจจะเป็นไปได้ว่าบาง state อาจจะมี copy มารวมกันก็รวมกันเป็น copy เดียว

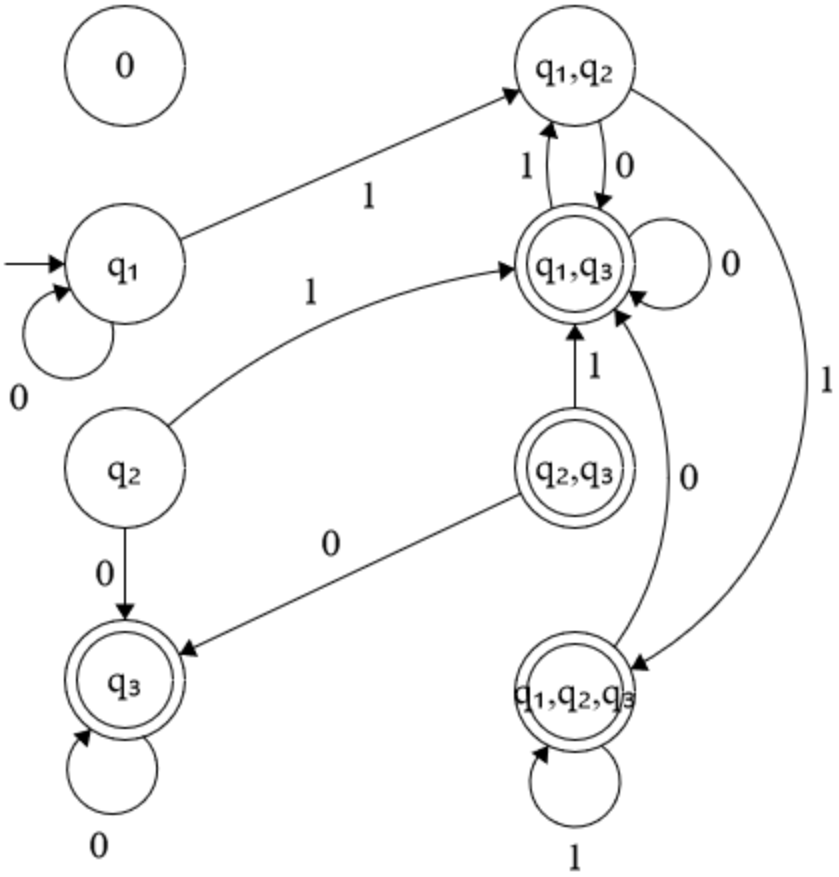
ดังนั้น state ทั้งหมดที่เป็นไปได้จะมี

$\{\emptyset, \{q_1\}, \{q_2\}, \{q_3\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$


0,1
0
q1
1
q2
0,1
q3


0
q1
q2
q3
q1,q2
q1,q3
q2,q3
q1,q2,q3


0
q1,q2
1
1
0
0
q1
q1,q3
0
1
1
1
q2
q2,q3
0
0
0
q3
q1,q2,q3
0
1


0
q1,q2
1
1
0
q1
q1,q3
0
1
0
1
q2
q2,q3
0
0
0
q3
q1,q2,q3
0
1

ให้ $\mathrm{N} = (Q, \Sigma, \delta, q_0, F)$ เป็น NFA

เราจะสร้าง DFA $\mathrm{M} = (Q', \Sigma, \delta', q_0', F')$ ที่ recognize ภาษาเดียวกัน

กรณีง่าย

N ไม่มีลูกศรที่มี input เป็น ϵ (สอดคล้องกับที่ทำเมื่อกี้)

$$Q' = P(Q)$$

ถ้า N เป็น state $q \in Q$ และรับ input symbol $a \in \Sigma$, N อาจจะย้ายไปยัง state ต่างๆ ใน $\delta(q, a)$

M จะทำตัวว่าเป็นหลายๆ state ใน N นั่นคือ state ของ M เป็นบาง state $R \subseteq Q$ เมื่อ a เป็น input state ถัดไปที่จะเป็นไปได้คือ

$$\delta'(R, a) = \bigcup_{q \in R} \delta(q, a)$$

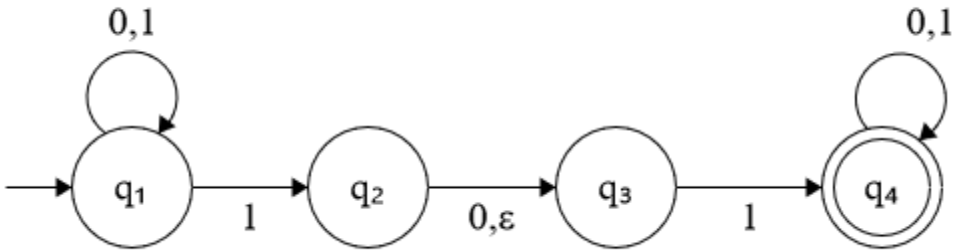
$$q_0' = \{q_0\}$$

$$F' = \{R \in Q' | R \text{ มี } accept\ state \text{ ของ } N\}$$

หากมี ϵ จะจัดการอย่างไร

เนื่องจากเมื่อมี ϵ อยู่บนลูกศร N จะสามารถย้ายได้อย่างอิสระไปตามลูกศรโดยที่ไม่ต้องรับ input

ตกลงกันก่อนว่า เดินแล้วค่อย copy ตัวเอง



จะนิยาม สำหรับ state $q \in Q$,ให้ $D(q)$ เป็นเซตของ state ทั้งหมดที่ไปได้จาก q ผ่านทางลูกศร ϵ เป็นจำนวน 0 อันเป็นต้นไป ดังนั้นถ้าอยู่ที่ q แล้วมันจะสามารถเคลื่อนที่อย่างอิสระไป state ใดๆ ก็ได้ใน $D(q)$

D(q) มันไม่ได้อยู่ใน definition ของ NFAs มันอยู่ในการตีความของเรา

ทีนี้พิจารณา M ที่ start $R \in Q'$ ที่ Q'เป็น powerset ของ Q

นิยาม $E(R)$เป็นเซตของ state ที่ไปได้ถึงจาก state ใน R โดยผ่านทางลูกศร ϵ เป็นจำนวน 0 อันเป็นต้นไป

นั่นคือ $E(R) = \{\text{q|q สามารถไปได้ถึงจาก state ใน R}$
โดยผ่านทางลูกศร ϵ เป็นจำนวน 0 อันเป็นต้นไป$\}$

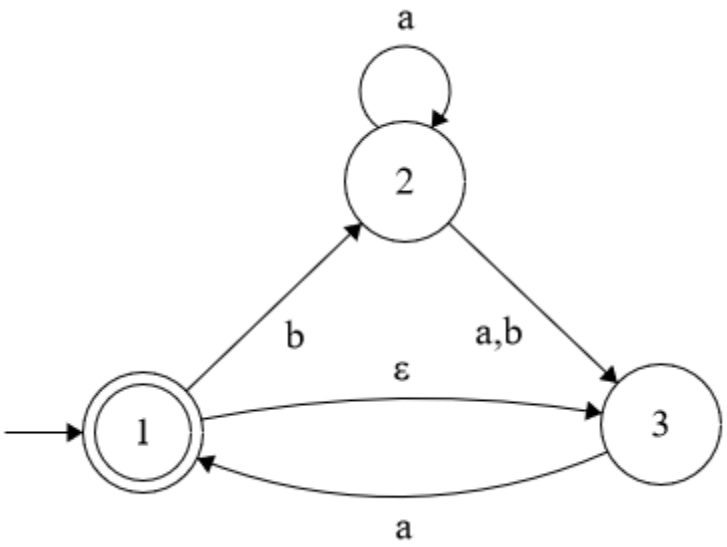
เมื่อนำมารวมกันจะได้ว่า

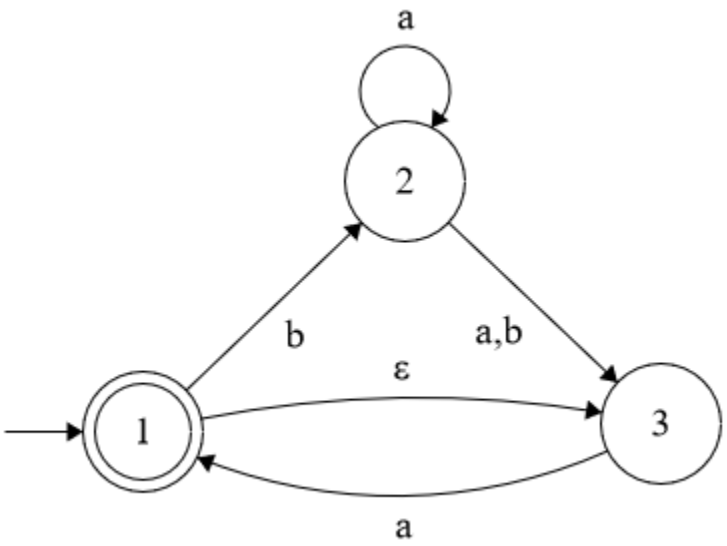
$$\delta'(R, a) = \cup_{q \in R} E(\delta(q, \text{a}))$$

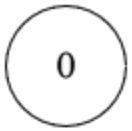
Fix จุดเริ่มต้น start states = $q_0' = E(\{q_0\})$

แปลง NFA ให้เป็น DFA

ข้อตกลงรับก่อนแล้วค่อยเดินตาม ϵ




a
2
b
a,b
ε
1
3
a


0


{1}


{2}

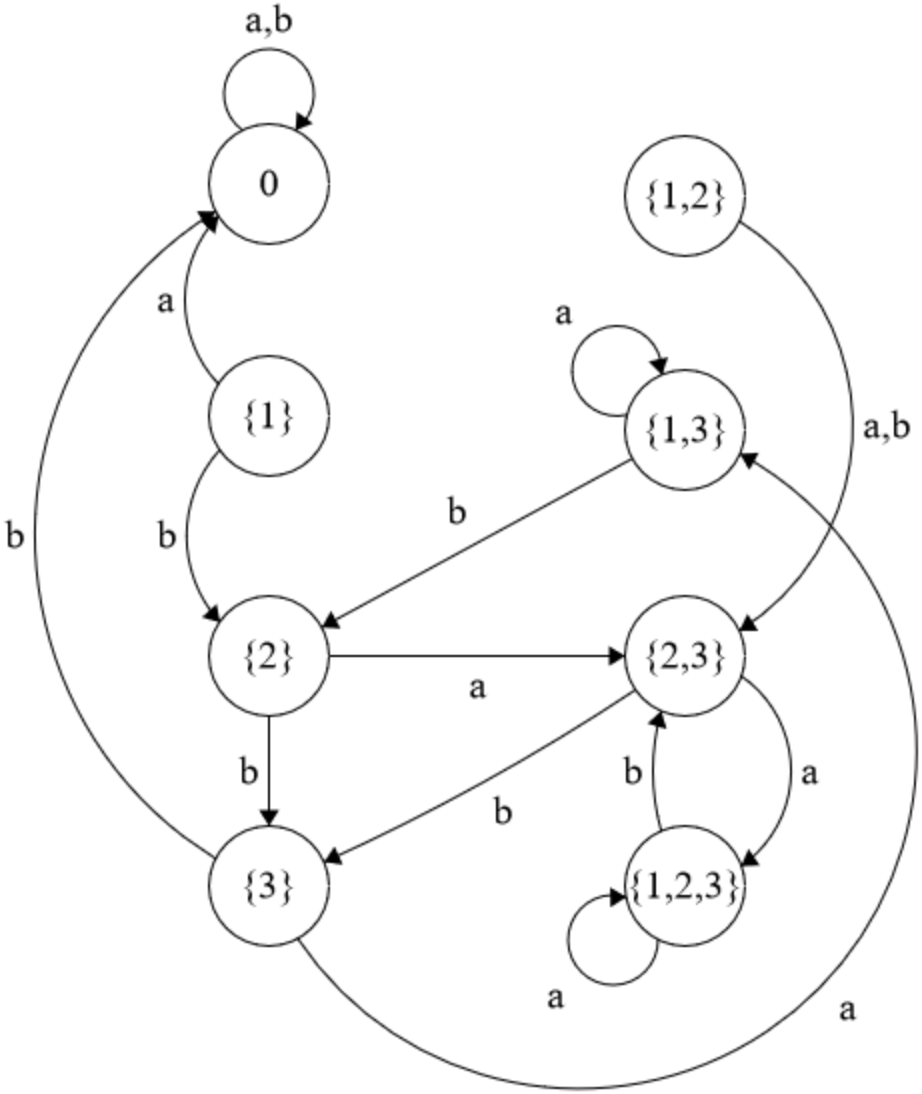

{3}


{1,2}

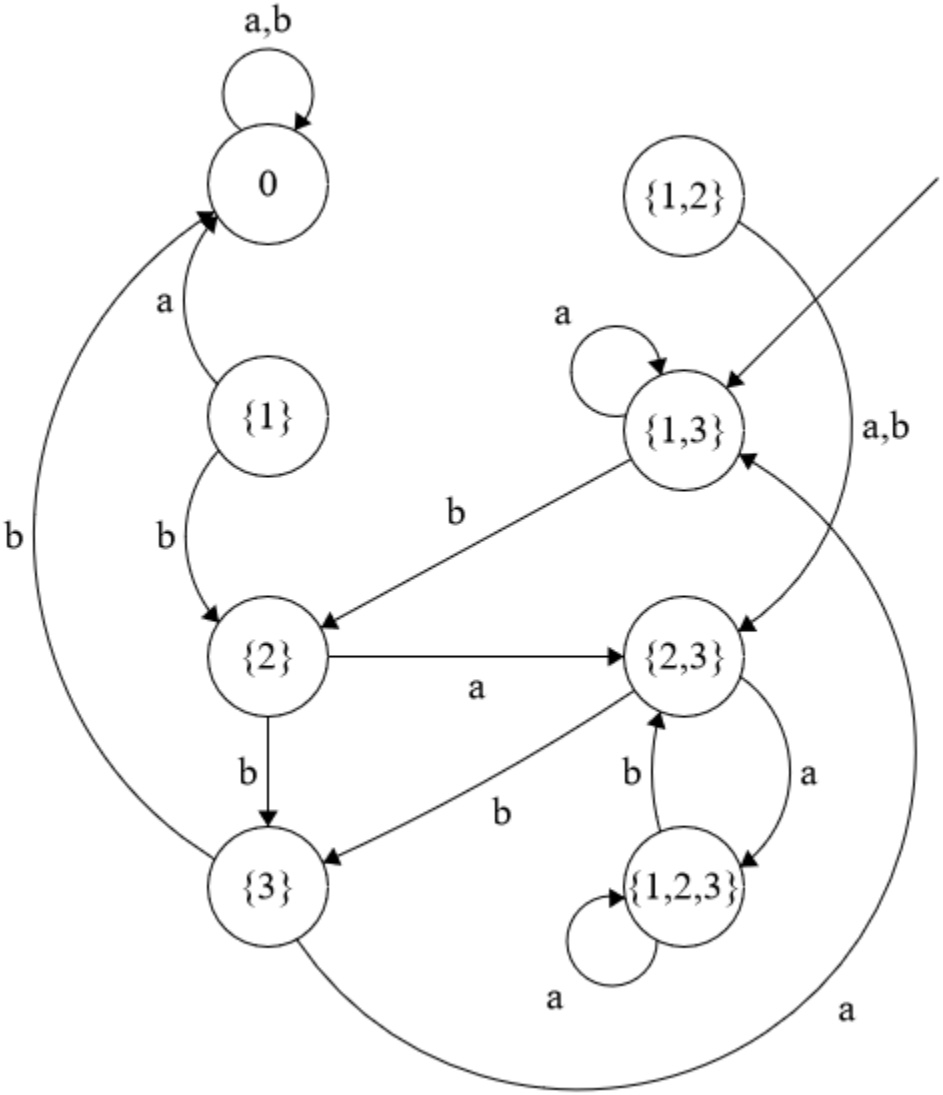

{1,3}

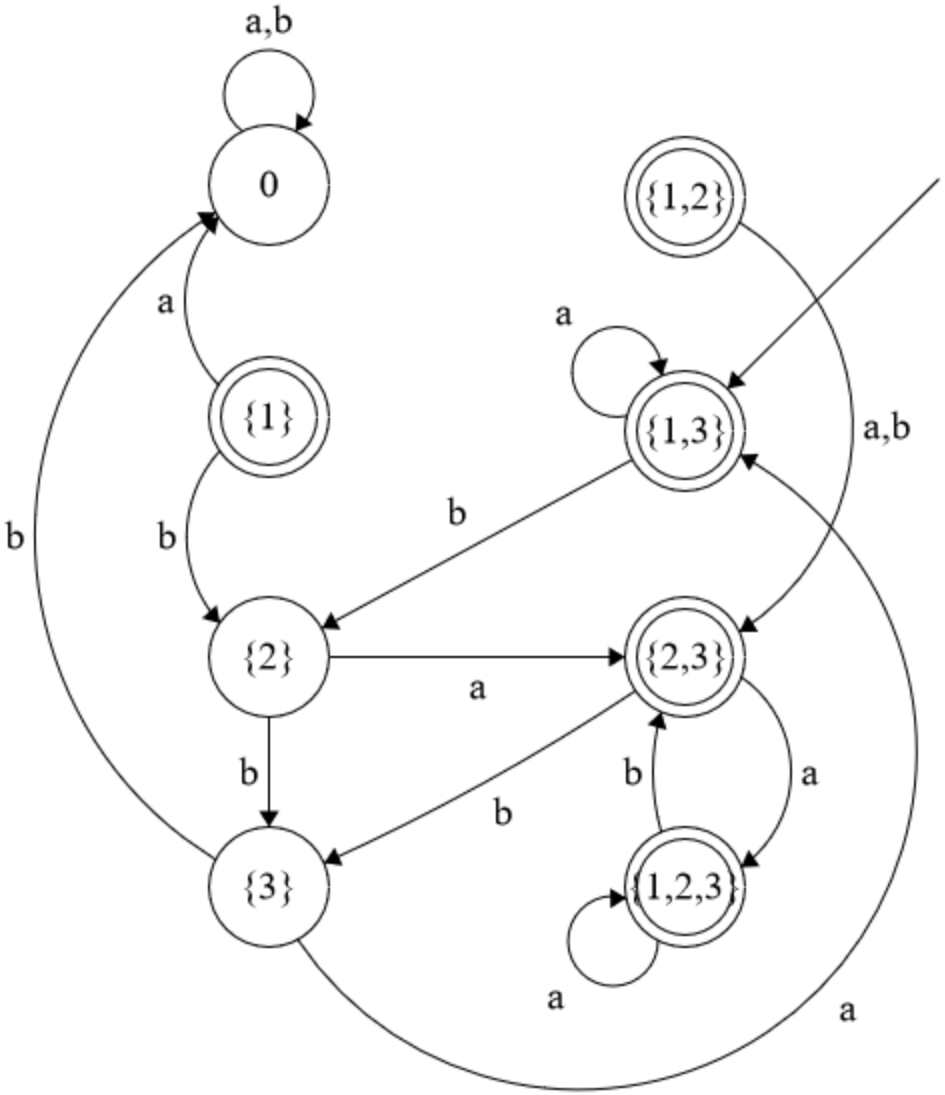

{2,3}


{1,2,3}


a,b
0
{1,2}
a
a
{1}
{1,3}
a,b
b
b
b
{2}
{2,3}
a
b
b
b
a
b
{3}
{1,2,3}
a
a

- Start state




a,b
0
{1,2}
a
a
{1}
{1,3}
a,b
b
b
b
{2}
{2,3}
a
b
b
b
a
{3}
{1,2,3}
a
a

จากที่เราสร้างให้ดู

เนื่องจากในแต่ละจุดที่ N ทำงาน M maintain set ของ state ทั้งหมด ที่ N สามารถไปอยู่ได้ M ก็จะ simulate N ได้อย่างถูกต้อง

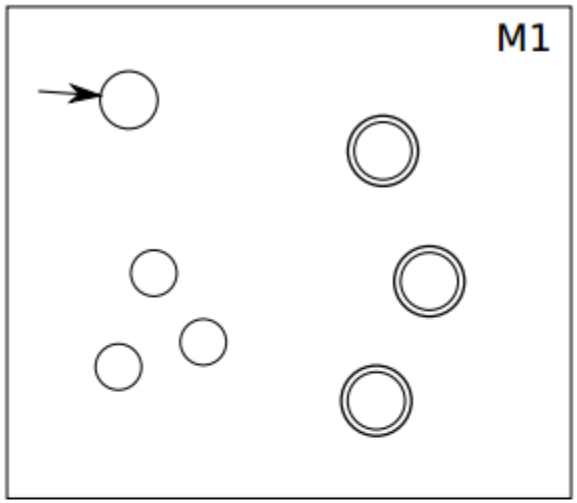
จากที่เราทำได้นี้ทำให้เราได้คุณสมบัติที่กว้างขึ้นของ regular language

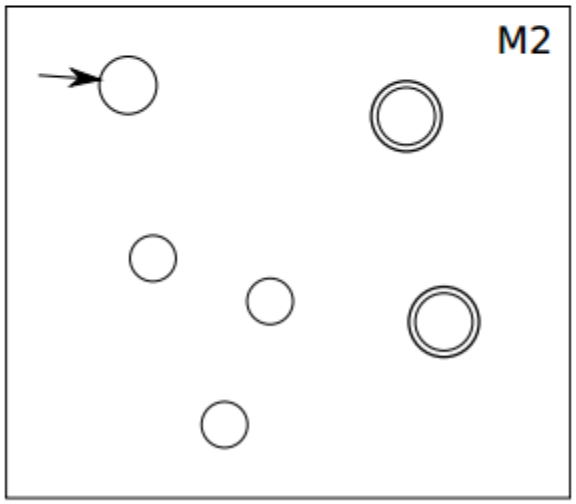
A language is regular iff some nondeterministic finite automaton recognizes it

ทำแบบนี้ได้แล้วมีข้อดีอย่างไร

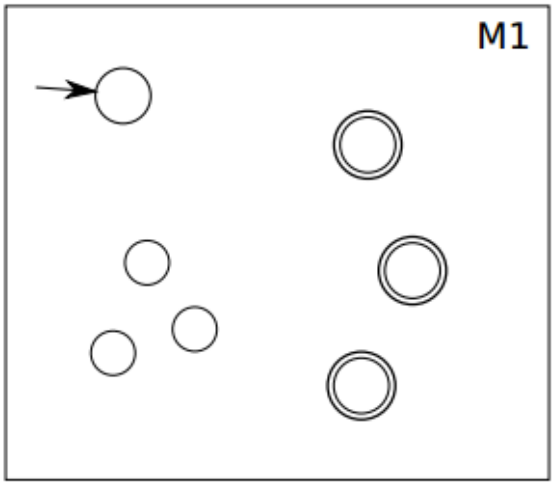
เนื่องจาก NFA-DFA equivalence ทำให้เรา ไม่ต้องสร้าง deterministic finite automata สำหรับแสดงว่า regular operations (concatenation union star) มีคุณสมบัติปิด เราก็สร้าง nondeterministic finite automata แทน

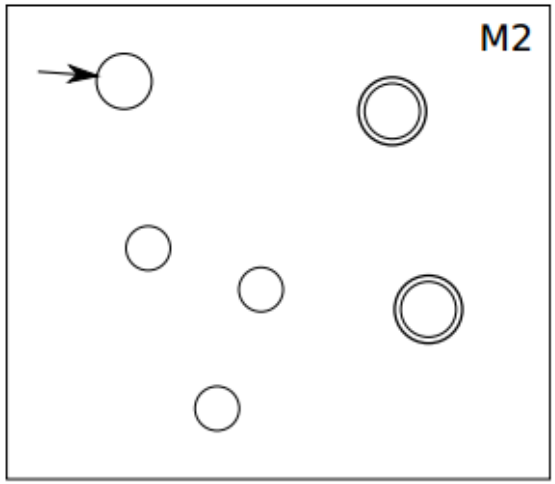
Union

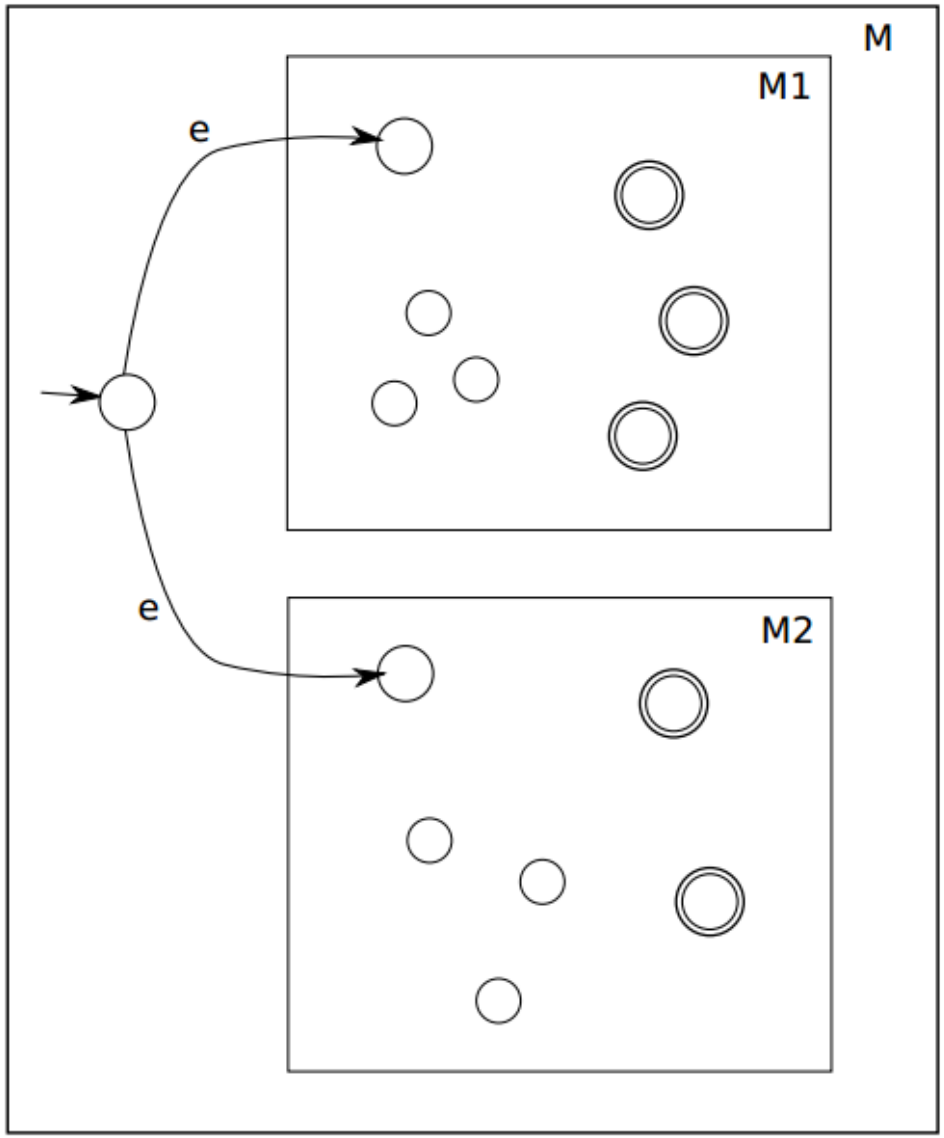




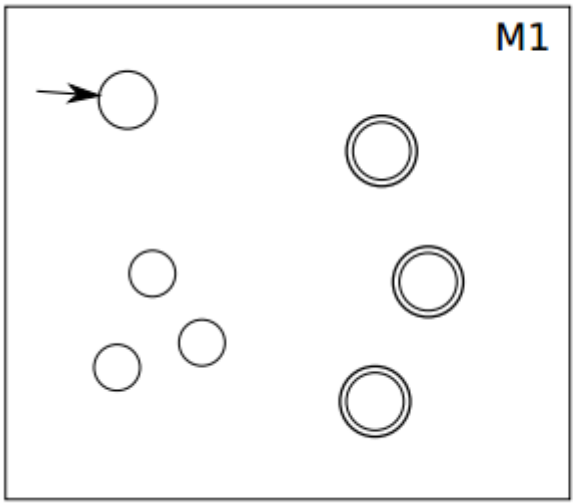
Union

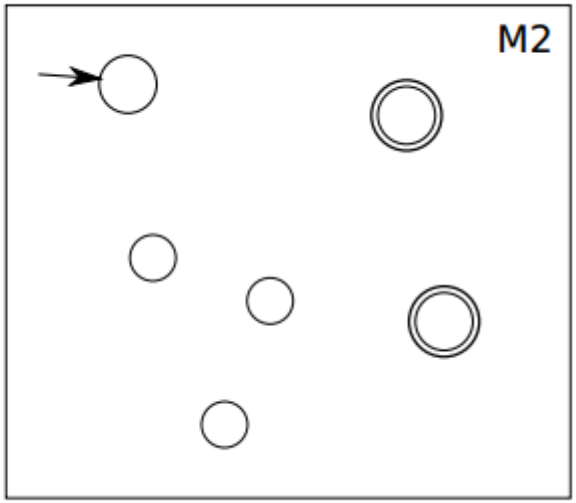




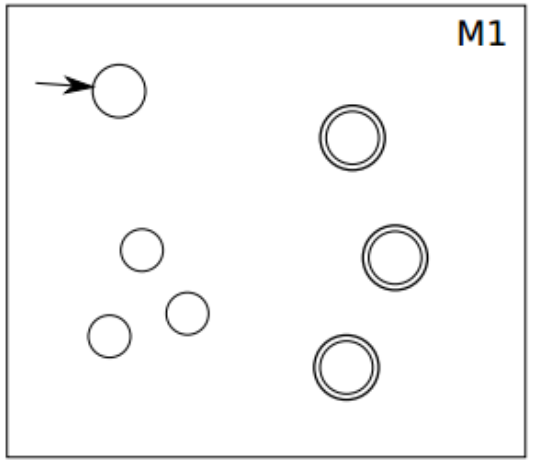


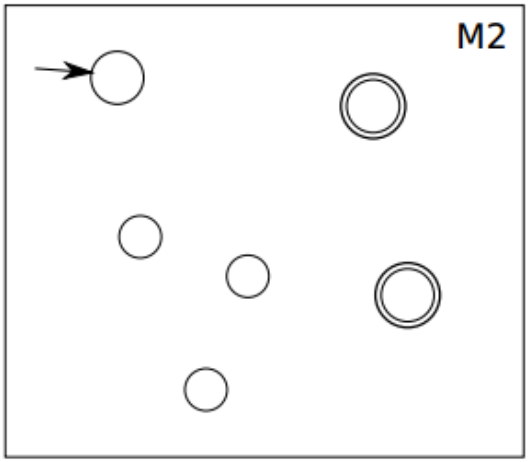
Concatenation

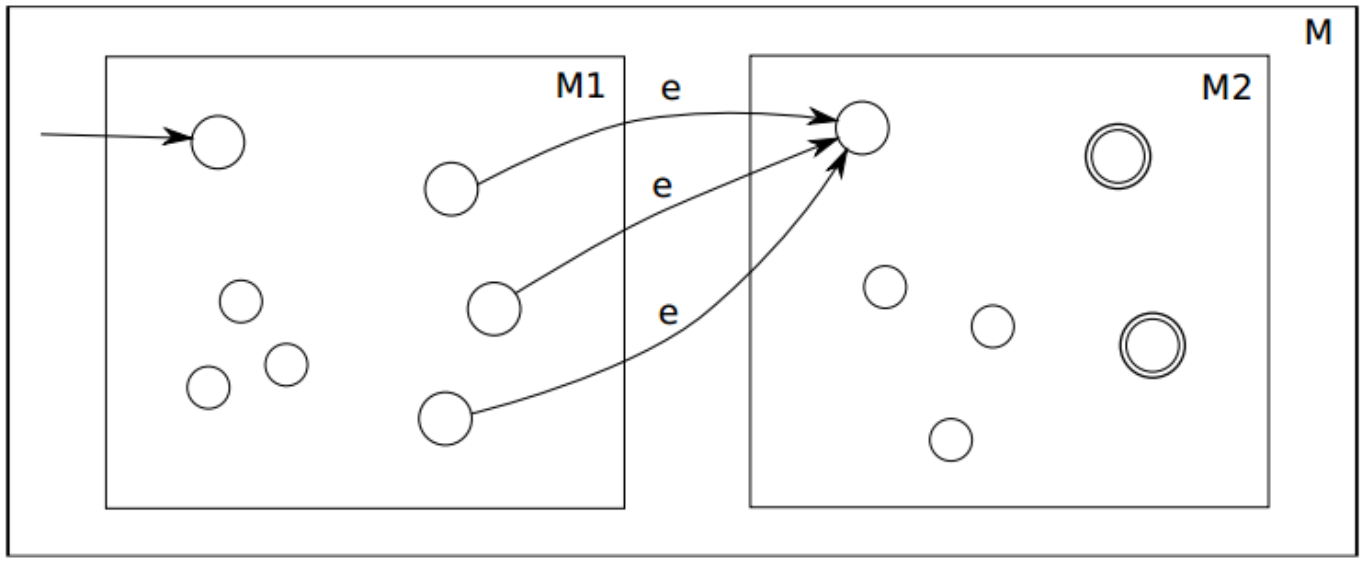




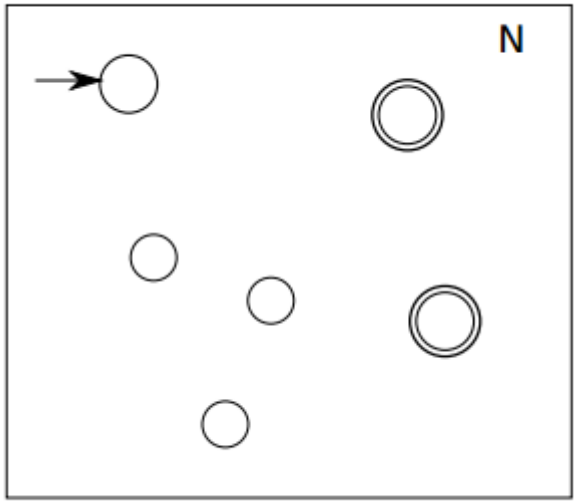
Concatenation







star



star