204753

Theory of computation

Lecture 2: Finite State Machine

Intro

Theory of computation จะเริ่มต้นด้วยคำถามที่ว่า "อะไรคือคอมพิวเตอร์" (ไม่ได้หมายถึงเครื่องที่เราใช้อยู่นะ)

เราจะพิจารณา เครื่องคอมพิวเตอร์ในอุดมคติ ซึ่งจะเรียกว่า **computational model**

เราจะใช้ computational model หลายแบบขึ้นกับ feature ที่เราสนใจ โดยเราจะเริ่มต้นด้วย model ที่พื้นฐานที่สุด finite state machine หรือ finite automata

Automata

คำว่า **Automata** มาคำในจากภาษากรีก "αὐτόματα“ ซึ่งหมายความว่า ทำงานได้ด้วยตัวเอง

Automaton (เอกพจน์, Automata พหูพจน์) หมายถึงรูปแบบนามธรรมของอุปกรณ์คำนวณที่ทำงานแบบอัตโนมัติตามลำดับของการดำเนินการที่กำหนดไว้ก่อน

Automaton ที่มีจำนวนสถานะ (State) จำกัดจะเรียกว่า **Finite Automaton(FA)** หรือ **Finite State Machine(FSM)**

Finite automata

Finite automata เป็นโมเดลที่ดีสำหรับคอมพิวเตอร์ที่มีหน่วยความจำที่จำกัด

คำถามคือ computer ที่มีหน่วยความจำน้อยมากๆ ทำอะไรได้บ้าง?

ในชีวิตประจำวัน เราได้ใช้คอมพิวเตอร์แบบนี้อยู่ตลอด

พวกเครื่องใช้ไฟฟ้า

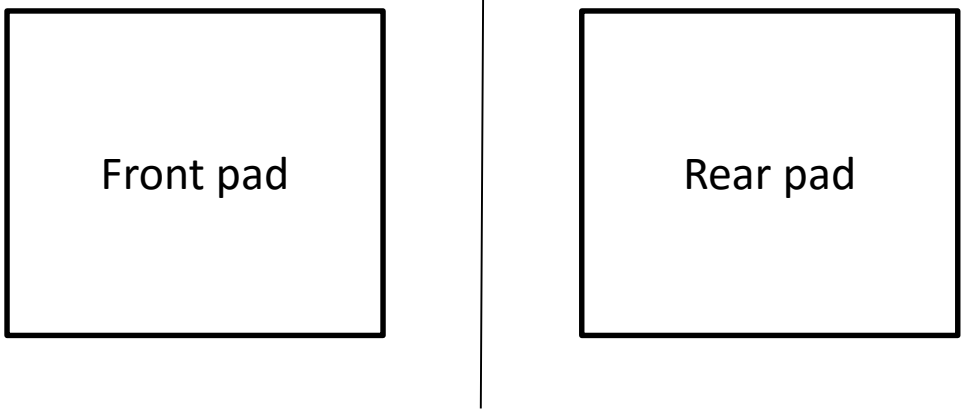
ตัวอย่างประตูอัตโนมัติ

ตัวควบคุมประตูอัตโนมัติ(เป็นบานพับ) เป็นตัวอย่างที่เห็นได้ชัดอย่างหนึ่ง

หลักการทำงานของมัน

ประตูจะเปิดเมื่อมันตรวจพบว่ามีคนยืนอยู่

ประตูอัตโนมัติจะมีเบาะด้านหน้าเพื่อตรวจว่ามีคนเดินผ่านมาทางเข้า และจะมีเบาะอีกอันด้านหลังทางเข้าเพื่อที่ตัวควบคุมจะเปิดประตูค้างไว้นานพอที่คนจะผ่านไปได้ และไม่ชนคนหากมีคนยืนด้านหลังถ้ามันเปิด



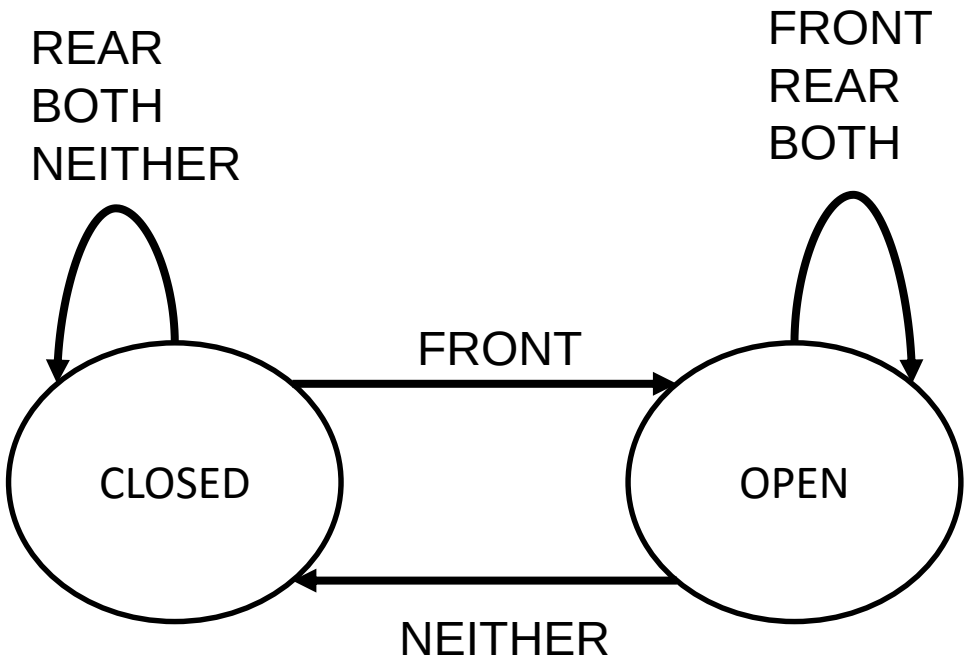
ในตัวอย่างนี้ตัวควบคุมจะอยู่ใน**สถานะ(state)** “OPEN” หรือ “CLOSED”
เราพบว่ามี**ข้อมูลเข้า(input)**ที่เป็นไปได้ 4 อย่างซึ่ง state ของเครื่องจะเปลี่ยน(หรือเหมือนเดิม) หลังจากได้รับ input

FRONT : หมายถึงคนยืนหนึ่งคนที่ Front pad

REAR : หมายถึงคนยืนหนึ่งที่ Rear pad

BOTH : หมายถึงคนยืนที่ pad ทั้งสอง

NEITHER : หมายถึงไม่มีใครยืนที่ pad ใดเลย



State Diagram สำหรับตัวควบคุมประตูอัตโนมัติ

State transition table ของตัวควบคุมประตูอัตโนมัติ

Input signal

state	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

ตัวควบคุมจะเปลี่ยนจากสถานะหนึ่งไปอีกสถานะ ขึ้นกับว่า input ที่ได้รับเป็นอะไร

- เมื่อประตูอยู่ในสถานะ CLOSED แล้วได้รับ input เป็น NEITHER หรือ REAR มันจะยังคงอยู่ในสถานะ CLOSED ถ้า input เป็น BOTH มันจะยังคง CLOSED เพราะว่าการเปิดประตูเสี่ยงที่จะไปชนคนที่ยืนอยู่ที่ Rear pad แต่ถ้า input เป็น FRONT มันจะย้ายสถานะเป็น OPEN
- ในสถานะ OPEN นี้ถ้าได้รับ input เป็น FRONT REAR BOTH จะยังคงเปิด แต่เมื่อได้รับ input เป็น NEITHER จะเปลี่ยนเป็นสถานะเป็น CLOSED

ลองดู

Input	สถานะ
เริ่มต้น	CLOSED
FRONT	
REAR	
NEITHER	
FRONT	
BOTH	
NEITHER	
REAR	
NEITHER	

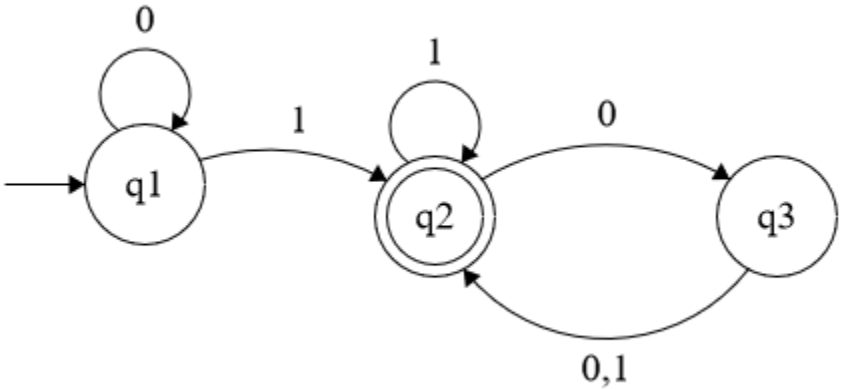
เมื่อพิจารณาดูแล้วประตูอัตโนมัตินี้เป็น Finite automata เราจะมองตัวควบคุมเป็น computer ที่มีหน่วยความจำ 1 bit สามารถจดจำสถานะของตัวควบคุมได้ 2 สถานะ

ตัวอย่างอุปกรณ์อื่นที่ตัวควบคุมมีหน่วยความจำที่ใหญ่กว่าได้แก่ ลิฟต์ ตัวควบคุมของลิฟต์นั้นสถานะอาจจะหมายถึงชั้นที่ลิฟต์อยู่ และ input อาจจะเป็นสัญญาณจากปุ่มกด computer นี้อาจจะใช้หลาย bit ในการเก็บข้อมูลเหล่านี้

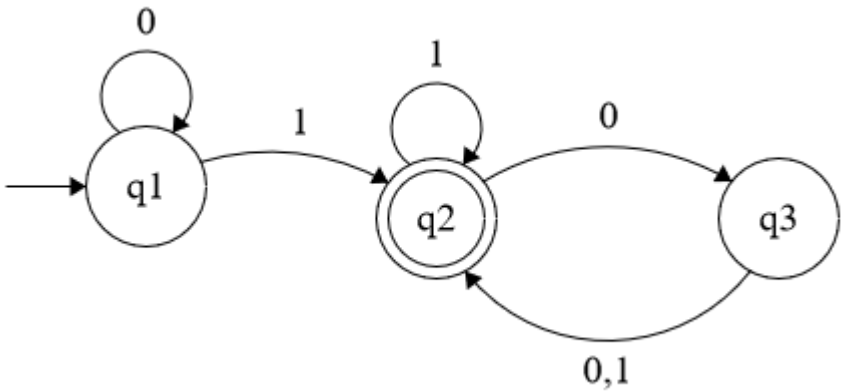
ตัวควบคุมอื่นๆ ของเครื่องใช้ในบ้านเช่นเครื่องล้างจาน เครื่องซักผ้า ก็เป็นตัวอย่างของ computer ที่มีหน่วยความจำจำกัด

เราจะพิจารณา finite automata ในมุมมองทางคณิตศาสตร์

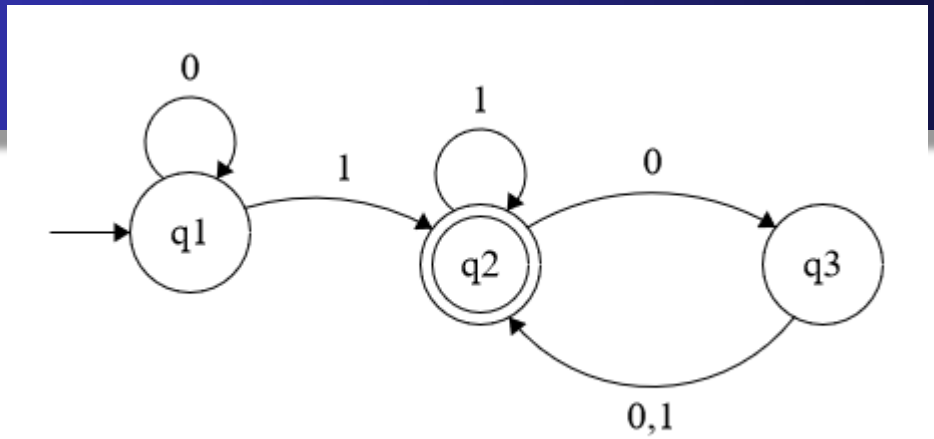
โดยเราจะให้นิยามที่ชัดเจนของ finite automata, คำศัพท์ต่างๆ ที่เกี่ยวข้องสำหรับอธิบายและดำเนินการของ finite automata และผลลัพธ์ทางทฤษฎีที่อธิบายพลังและข้อจำกัด



Finite automaton: M_1 ที่มี 3 state



รูปนี้เรียกว่า **state diagram** ของ M_1 มี 3 state มีชื่อ q_1,q_2,q_3
start state q_1 รู้ได้จากเป็น state ที่มีลูกศรชี้เข้าที่ไม่มีต้นทาง
accept state q_2 รู้ได้จากเป็น state ที่มีวงสองวงซ้อนกัน
ลูกศรที่ชี้จาก state หนึ่งไปอีก state หนึ่ง เรียกว่า **transitions**

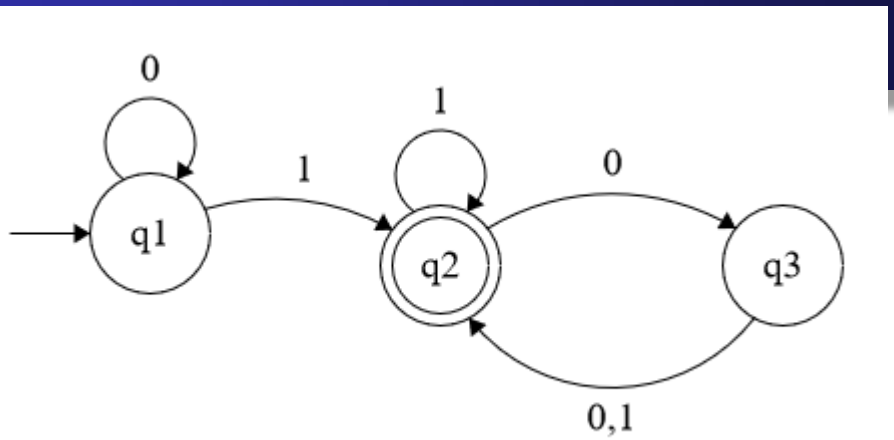


เมื่อ automaton ได้รับ input string เช่น 1101 มันจะดำเนินการตาม string และให้ output ออกมา โดย output เป็นได้ accept หรือ reject

กระบวนการทำงานของ M_1 จะเริ่มจาก start state q_1

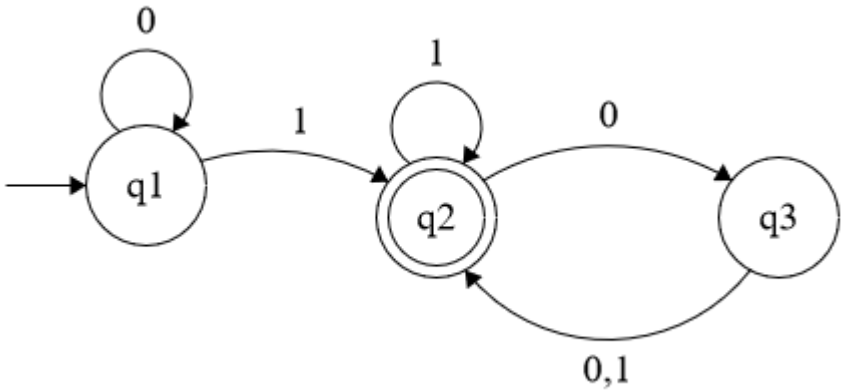
automaton รับ symbols จาก input string ทีละตัวจากซ้ายไปขวา หลังจากอ่านแต่ละ symbol M_1 จะย้ายจาก state หนึ่งไปอีก state หนึ่งตาม transition ที่ตรงกับ symbol

เมื่ออ่าน symbol ตัวสุดท้าย M_1 จะให้ output โดยถ้าหยุดที่ accept state จะให้ output เป็น accept นอกนั้น reject



เมื่อรับ input string เป็น 1101 ให้กับ M_1 จะทำงานดังนี้

1. เริ่มที่ q_1
2. อ่าน 1, เปลี่ยน transition จาก q_1 ไป q_2
3. อ่าน 1, เปลี่ยน transition จาก q_2 ไป q_2
4. อ่าน 0, เปลี่ยน transition จาก q_2 ไป q_3
5. อ่าน 1, เปลี่ยน transition จาก q_3 ไป q_2
6. Accept เพราะว่า M_1 อยู่ใน accept state q_2 เมื่อ input หมด

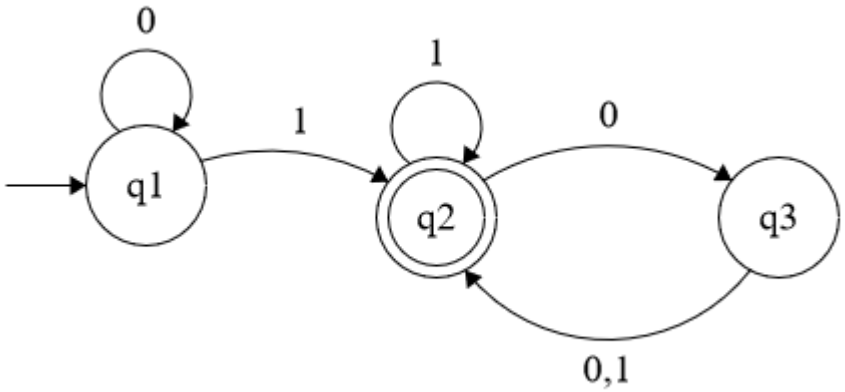


ทดลองกับ machine นี้ ด้วย input ต่อไปนี้

1, 01, 11, 010101010101

00, 10, 110, 11000

100, 1100, 110000



เราพบว่า M_1 accept string ที่ลงท้ายด้วย 1

นอกจากนี้เราพบว่า M_1 accept string ที่ลงท้ายด้วย 0 เป็นจำนวนคู่ซึ่งตามด้วย 1 อย่างน้อย 1 ตัว

เราพบว่า M_1 reject string เช่น 0, 10, 101000

เราจะอธิบายภาษาที่ M_1 accept ว่าอย่างไรดี

Formal Definition of a Finite Automata

- Formal definition ทำไมต้องมี
  - เพื่อความถูกต้อง precision
  - Notation

Formal Definition of a Finite Automata

A **finite automata** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta: Q \times \Sigma$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the set of **accept states**.

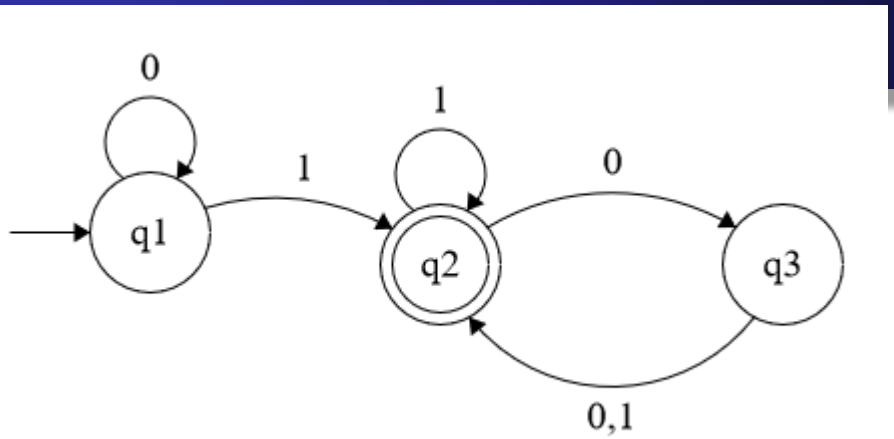
Transition function: δ

Transition function เรียกว่าเป็นกฎในการย้าย state

ตัวอย่างถ้าเราอยู่ที่ state q แล้วได้รับ input เช่น 1 มาแล้วเราจะย้ายไป state p:

$$\delta(q, 1) = p$$

ดังนั้น δ เป็นฟังก์ชันจาก **set ของ state** และ **set ของ input ที่เป็นไปได้** ไปยัง **set ของ state**



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

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ อธิบายได้ด้วย

	0	1
q1	q1	q2
q2	q3	q2
q3	q2	q2

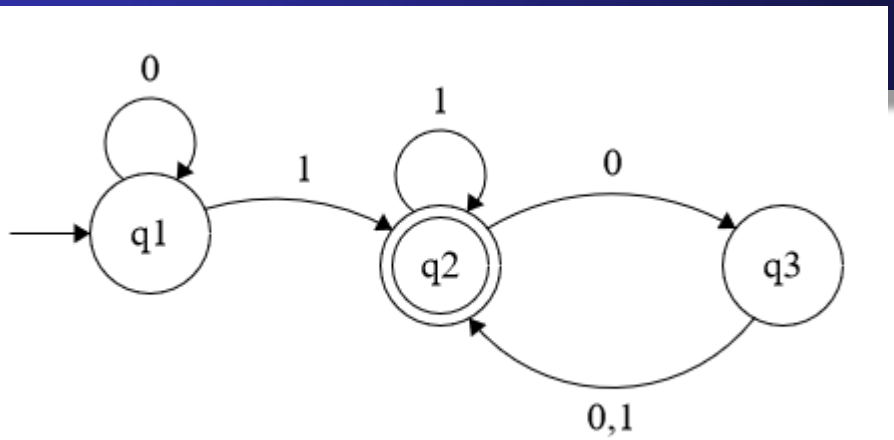
4. q_1 เป็น **start state**,
5. $F = \{q_2\}$

ถ้า A เป็นเซตของ string ทั้งหมดที่ **machine M accept** เราจะเรียก A ว่าเป็น **language of machine M** เขียนได้เป็น L(M) = A

บางทีเราจะบอกว่า **M recognizes A** หรือ **M accepts A**

Machine สามารถ accept ได้หลาย string แต่มันจะ recognizes ได้เพียง 1 language

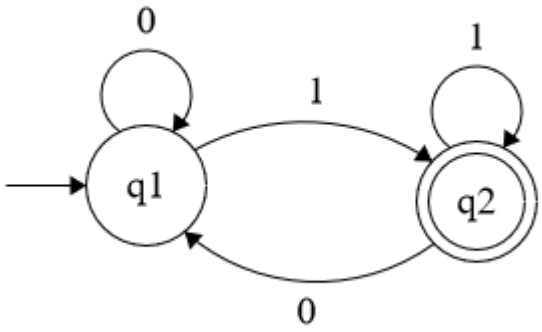
ถ้า machine accept no strings มันจะ recognize 1 language ที่ชื่อว่า **empty language** $\emptyset$



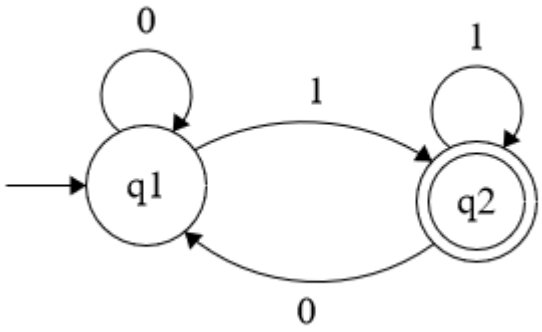
จากตัวอย่างนี้ให้

$$A = \{w | w \text{ contains at least one 1 and an even number of 0s follow the last 1}\}$$

จะได้ว่า $L(M_1) = A$ หรือ M_1 recognizes A

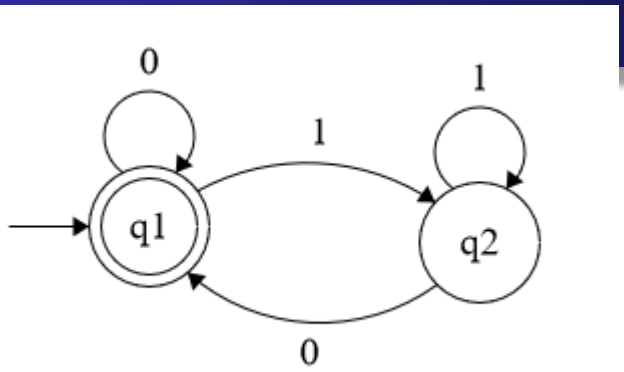


จงเขียน formal description ของ machine M_2



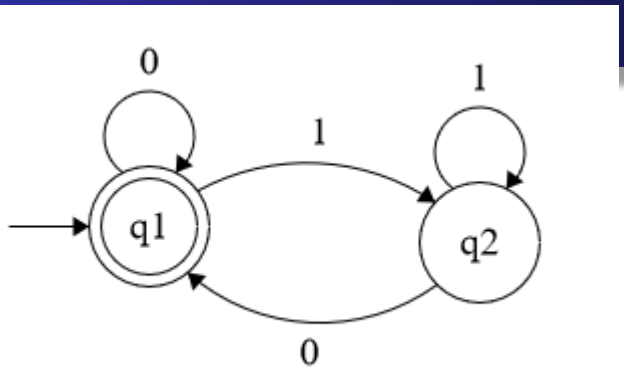
รูป state diagram ของ machine M_2

ภาษาอะไรที่ M_2 recognize



รูป state diagram ของ machine M_3

ภาษาอะไรที่ M_3 recognize



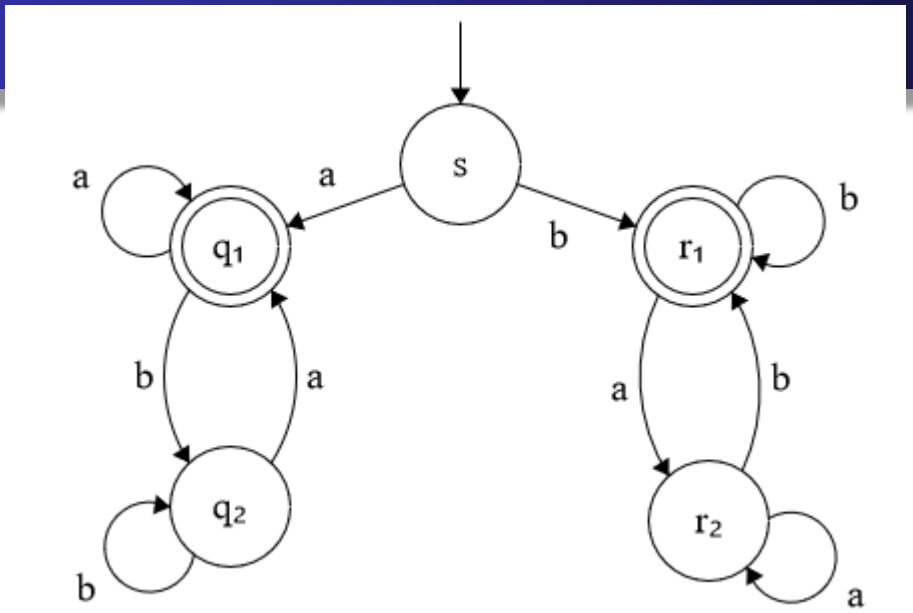
รูป state diagram ของ machine M_3

M_3 คล้ายกับ M_2 ต่างกันตรงตำแหน่งของ accept state

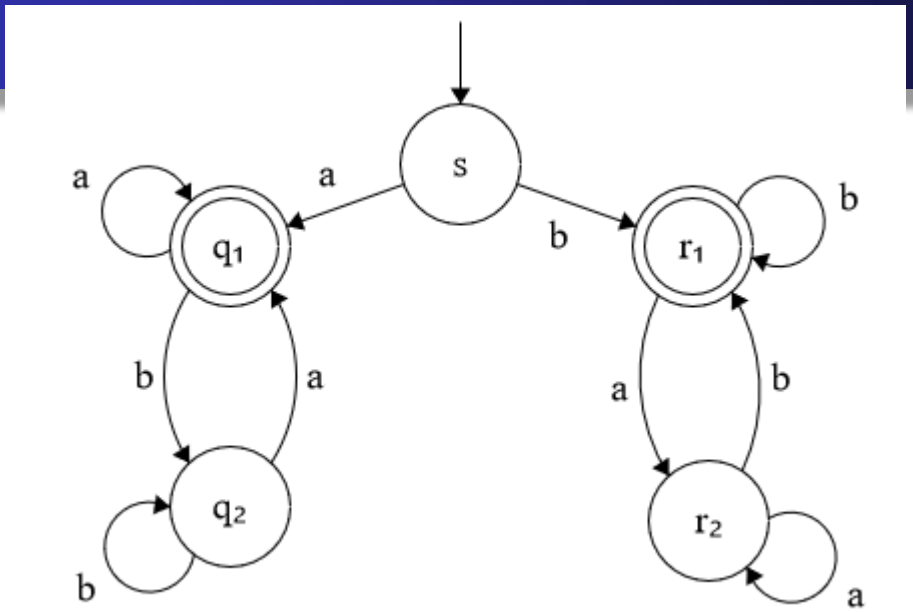
ข้อสังเกต เนื่องจาก start state เป็น accept state M_3 จึงสามารถรับ empty string ε ได้

Machine M_3 accepts empty string หรือ string ใดๆ ที่ลงท้ายด้วย 0

$$L(M_3) = \{w | w \text{ is the empty string } \varepsilon \text{ or ends in a } 0\}$$



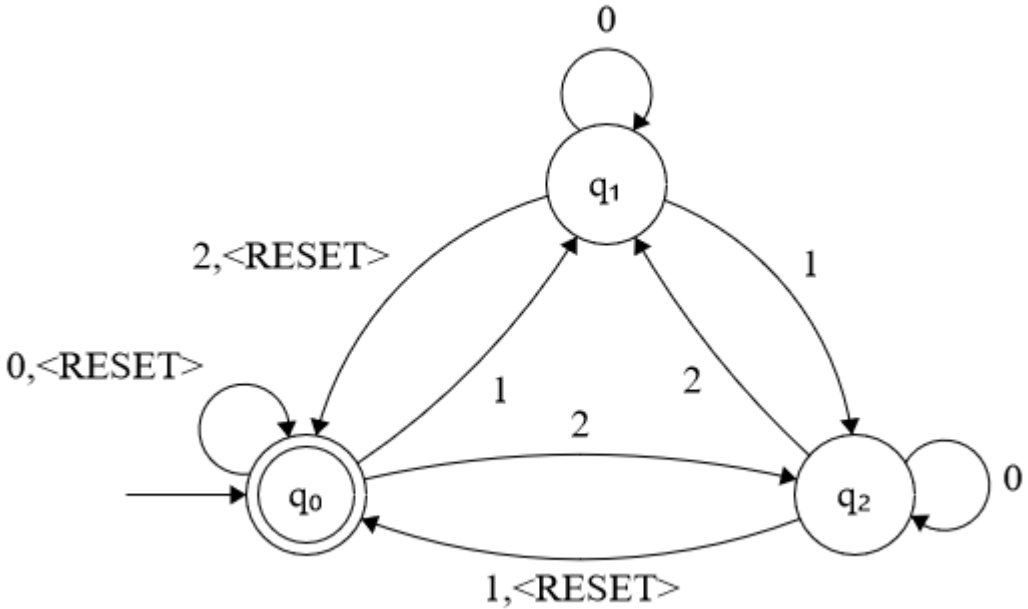
Machine M_4 recognize ภาษาอะไร



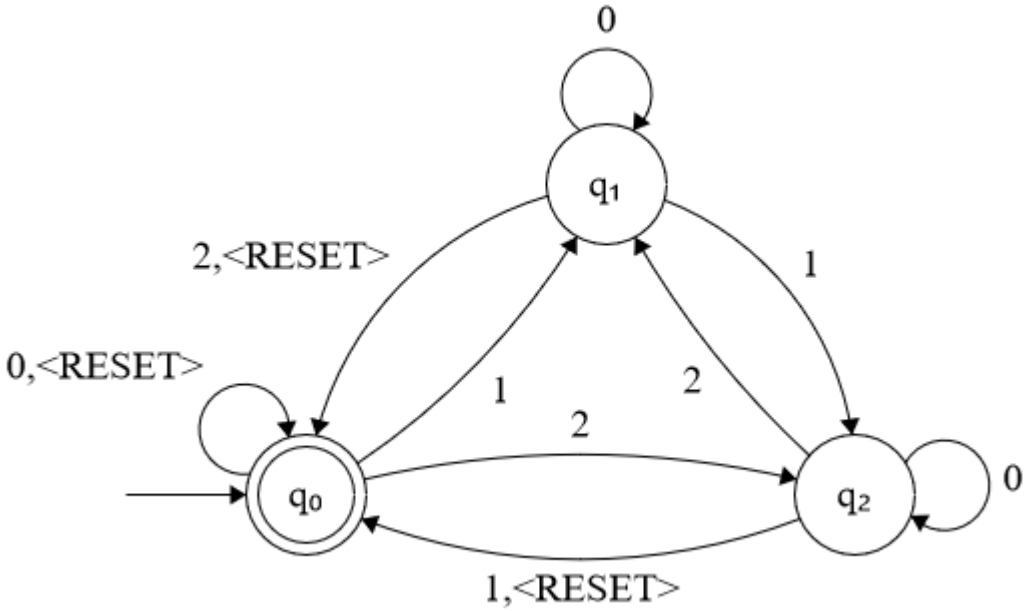
รูป state diagram ของ machine M_3

Machine M_4 มี 2 accept state q_1 และ r_1 ดำเนินการบนเซตตัวอักขระ $\Sigma = \{a, b\}$ เมื่อลองแทนค่าดูพบว่ามัน accept string a, b, aa, bb, aba แต่ reject string ab, ba, baa

เมื่อสังเกตดูดีๆ จะพบว่า M_4 accepts ทุก string ที่เริ่มต้นและลงท้ายด้วยอักขระตัวเดียวกัน



ภาษาอะไรที่ M_5 recognize



Machine M_5 มี input symbol 4 ตัว $\Sigma = \{< RESET >, 0,1,2\}$ โดยเรามองว่า <RESET> เป็น 1 symbol

Machine M_5 จะนับเอาของผลรวมของ input ที่เป็นตัวเลข modulo 3 และเมื่อไรที่รับ <RESET> จะเคลียร์ตัวนับเป็น 0 ใหม่

มันจะ accept ผลรวมที่เป็น 0, modulo 3(หรือผลรวมที่เป็นจำนวนเท่าของ 3)

การอธิบาย finite automaton ด้วย state diagram บางครั้งทำไม่ได้ในบางกรณี ส่วนใหญ่จะเกิดขึ้นเมื่อ diagram ใหญ่หรือมีเงื่อนไข อย่างเช่น M_5 ในกรณีนี้เราจะไปใช้ formal description แทน

สำหรับแต่ละ i >=1 ให้ A_i เป็นภาษาของทุก strings ที่ผลรวมของจำนวนเป็นจำนวนเท่าของ i ยกเว้นผลรวมจะถูกปรับเป็น 0 เมื่อได้รับ <RESET>

สำหรับแต่ละ A_i เราให้ finite automaton B_i recognize A_i

ดังนั้น Machine B_i นิยามด้วย $B_i = \{Q_i, \Sigma, \delta_i, q_0, \{q_0\}\}$ เมื่อ Q_i เป็นเซตของ i state $\{q_0, q_1, \ldots, q_{i-1}\}$ และ จะให้ Transition function δ_i สำหรับแต่ละ j ถ้า B_i อยู่ใน q_j ผลรวมเป็น j ที่ mod i

สำหรับแต่ละ q_j ให้

$\delta_i(q_j, 0) = q_j$,

$\delta_i(q_j, 1) = q_k$ เมื่อ k=j+1 modulo i,

$\delta_i(q_j, 2) = q_k$ เมื่อ k=j+2 modulo i,

$\delta_i(q_j, < RESET >) = q_0$,

Formal definition of computation

ให้ M = $(Q, \Sigma, \delta, q_0, F)$ เป็น finite automaton

และให้ $w = w_1 w_2 \ldots w_n$ เป็น string ที่แต่ละ w_i เป็นสมาชิกของเซตอักขระ Σ แล้ว **M accept w** ถ้าลำดับของ state $r_0 r_1 \ldots r_n$ ใน Q ที่เกิดขึ้นมีเงื่อนไข 3 เงื่อนไขดังนี้

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for i= 0, 1, …, n-1 และ
3. $r_n \in F$

เงื่อนไขที่ 1 บอกว่า machine เริ่มต้นที่ start state

เงื่อนไขที่ 2 บอกว่า machine เปลี่ยนจาก state หนึ่งไปอีก state หนึ่ง ตาม transition function

เงื่อนไขที่ 3 บอกว่า machine accept input ของมันถ้ามันจบที่ accept state

นิยาม

เราจะบอกว่า **M recognize language A** ถ้า A={w| M accept w}

A language is called a **regular language** if some finite automaton recognizes it.

หากนำเอา machine M_5 มาเป็นตัวอย่าง ให้ w เป็น string

10<RESET>22<RESET>012

จะพบว่า M_5 accept w ตาม formal definition of computation เพราะว่าลำดับของ state ที่รับเข้ามาเมื่อคำนวณเป็น

$$q_0,q_1,q_1,q_0,q_2,q_1,q_0,q_0,q_1,q_0$$

ซึ่งสอดคล้องกับเงื่อนไขทั้ง 3

ภาษาของ machine M_5 คือ

$L(M_5)$ = {w| the sum of the symbols in w is 0 or modulo 3, except that <RESET> resets the count to 0}

ดังนั้น M_5 recognizes ภาษานี้ และเป็น regular language

Designing Finite Automata

- คิดว่าเราเป็น automaton
- ในแต่ละครั้ง รับ input ทีละตัว
- คิดว่าเราต้องจำอะไรเพื่อที่จะได้ตัดสินใจได้อย่างถูกต้อง (นั่นจะเป็น set ของ state)

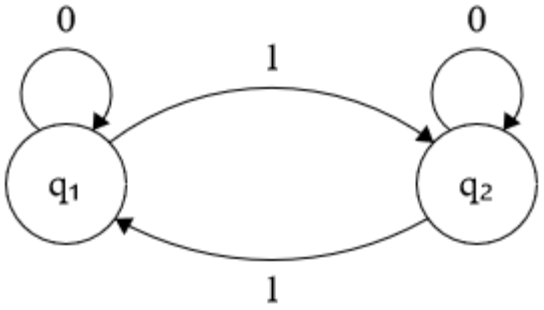
Designing Finite Automata

- ภาษาที่ประกอบไปด้วยทุก string ที่มี 1 เป็นจำนวนคี่

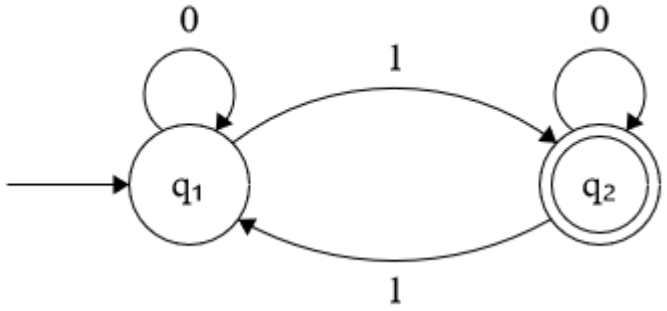
Designing Finite Automata

- ตัวอย่างสมมติว่ามีเซตของอักขระ {0,1} และภาษาประกอบด้วยทุก string ที่มี 1 จำนวนคี่ตัว
- เราจะมาออกแบบ finite automata E1 ที่ recognize ภาษานี้
- เราต้องคิดว่าเราจะเก็บหรือนับว่าตอนนี้ 1 มีจำนวนเป็นคี่หรือยังได้อย่างไร
- เราจำเป็นต้องเห็น string ทั้งสายก่อนไหม จริงๆไม่จำเป็นเราทำได้ง่ายๆ โดยเราจะจำว่าตอนนี้จำนวน 1 เป็นคี่หรือคู่ตัว ถ้าอ่าน symbol มาแล้วเจอ 1 เราก็จะเปลี่ยนสถานะ(คู่ไปคี่หรือคี่ไปคู่) แต่ถ้าอ่านเจอ 0 ก็ไม่เปลี่ยน
- ในการออกแบบ E1 เราก็จะดูว่าข้อมูลอะไรที่เราต้องจำ แล้วก็มาระบุว่ามีอะไรบ้างที่เป็นไปได้ ตัวอย่างนี้เราระบุได้
  - เป็นคู่มาก่อน
  - เป็นคี่มาก่อน

- เราจะกำหนด state ให้กับแต่ละสถานะที่เป็นไปได้ (q_1 เป็นคู่ q_2 เป็นคี่)
- จากนั้นจะมากำหนด transition จากการมองว่าวิธีการเปลี่ยนจากสถานะหนึ่งไปอีกสถานะหนึ่งเมื่อได้รับ symbol



- ขั้นต่อไประบุ start state ว่าถ้าไม่มี symbol หรือเป็น empty string จะอยู่ที่ state ไหน
- ตัวอย่างนี้ start state ที่สอดคล้องคือ q_1 เพราะว่า 0 เป็นจำนวนคู่
- ต่อไประบุ accept state



จงออกแบบ finite automata E2 ที่ recognize regular language ของทุก string ที่มี 001 เป็น substring ตัวอย่างเช่น 0001, 1001, 001, 101110101100101 ทุกตัวที่กล่าวมาอยู่ในภาษา แต่ 11, 000 ไม่ใช่

Regular operations

- Operation สำหรับดำเนินการกับ Language
- เป็น toolbox สำหรับ
  - สร้าง machine ที่ซับซ้อนขึ้น
  - เป็นการให้เหตุผลกับ class ของ machine

A Set and operations

Set จะมีคุณสมบัติปิดภายใต้การดำเนินการบางอย่างถ้าเอาสมาชิกของเซตมาดำเนินการภายใต้การดำเนินการนั้นแล้วผลลัพธ์ที่ได้ก็ยังอยู่ในเซต

ตัวอย่างเช่น เซตของจำนวนธรรมชาติ มีคุณสมบัติปิดภายใต้การคูณ

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

ตัวอย่างให้ Σ เป็นตัวอักษรภาษาอังกฤษตัวเล็ก 26 ตัว {a, b,…, z} ถ้า A={good, bad}และ B={boy, girl}

$$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$$
$$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$$
$$A^* = \{\epsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badbad}, \text{badgood}, \ldots\}$$

Irregular Language

ภาษาจะเป็น **irregular** ถ้าไม่มี finite automaton ที่ recognize มันได้

เราจะมาดูการดำเนินการทีละตัว เริ่มที่ union

ถ้า A_1 และ A_2 เป็น regular แล้ว $A_1 \cup A_2$ เป็น regular?

ลองคิดดูว่าถ้ารับ string มาตัวหนึ่งแล้ว จะเลือกใช้ machine ยังไงดี

ตัวอย่าง $A_1 \cup A_2$

สมมติว่าเรามี M1 ที่ recognize A_1 และ M2 ที่ recognize A_2 แล้ว อยากจะ recognize $A_1 \cup A_2$

ให้ M_1 accept string ที่ลงท้ายด้วย 1

ให้ design

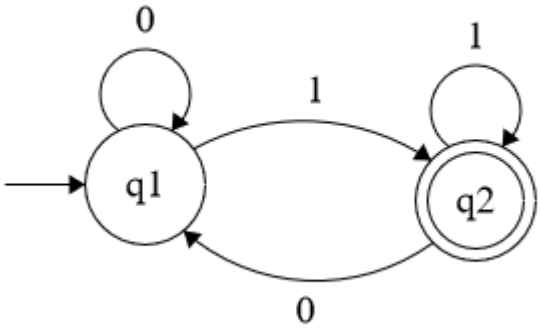
M_2 accept String ที่ลงท้ายด้วย 100

ให้ design

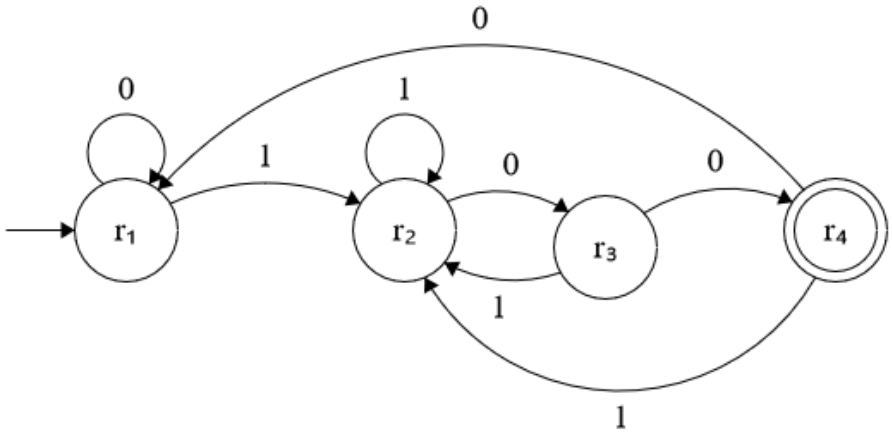
ตัวอย่าง $A_1 \cup A_2$

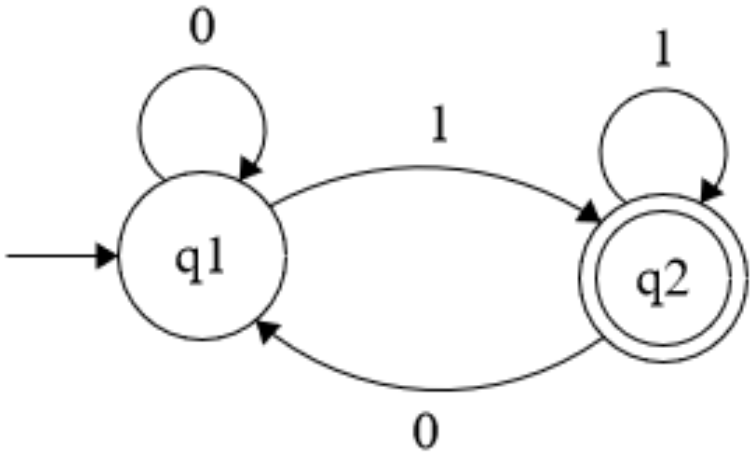
สมมติว่าเรามี M1 ที่ recognize A_1 และ M2 ที่ recognize A_2 แล้วอยากจะ recognize $A_1 \cup A_2$

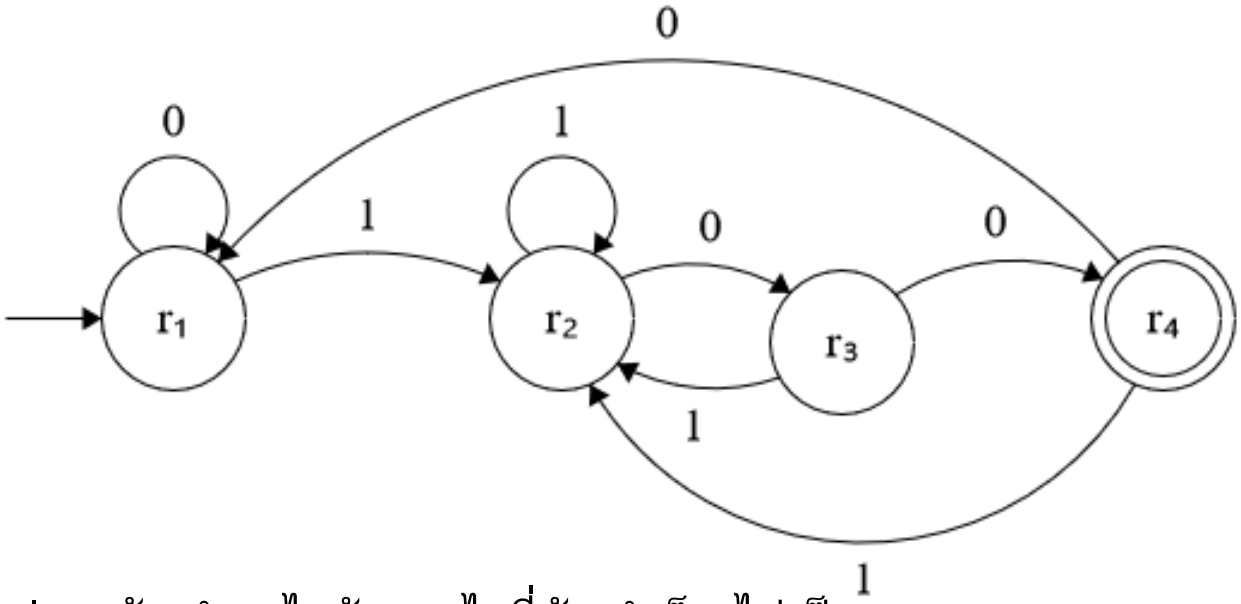
ให้ M_1 accept string ที่ลงท้ายด้วย 1



M_2 accept String ที่ลงท้ายด้วย 100



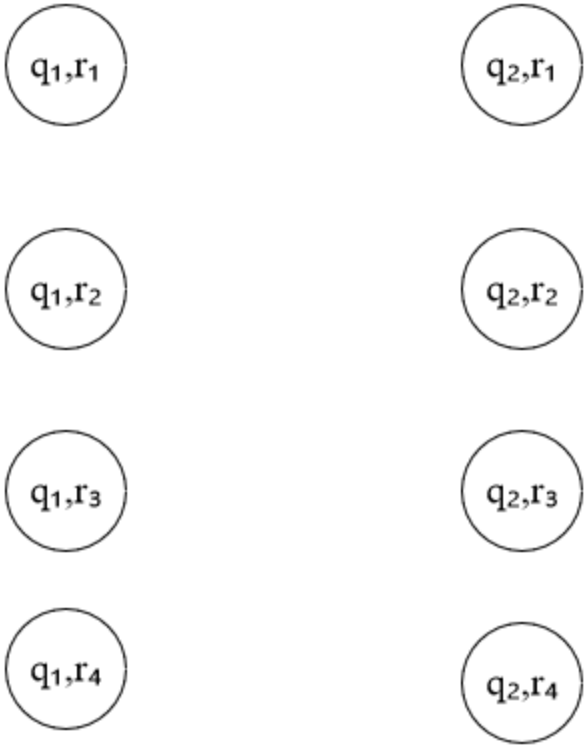


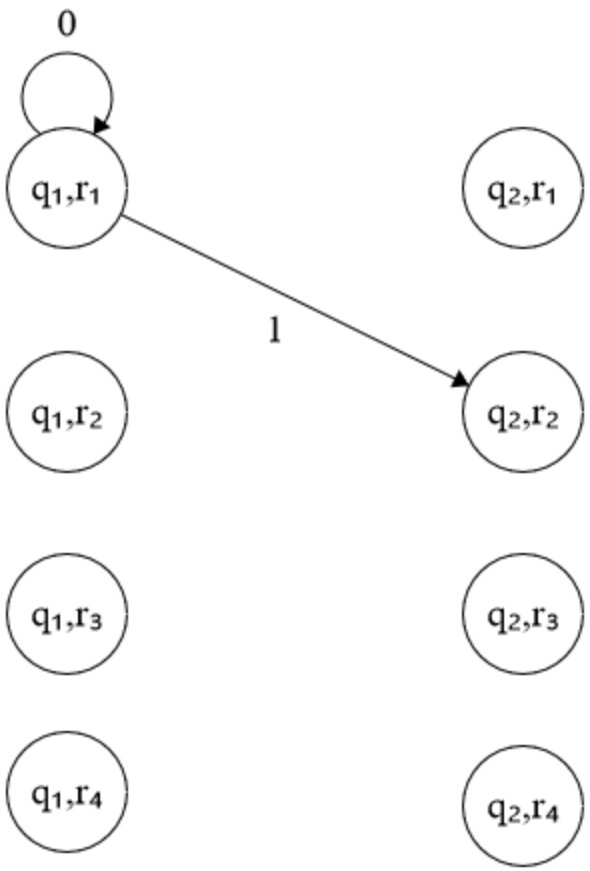


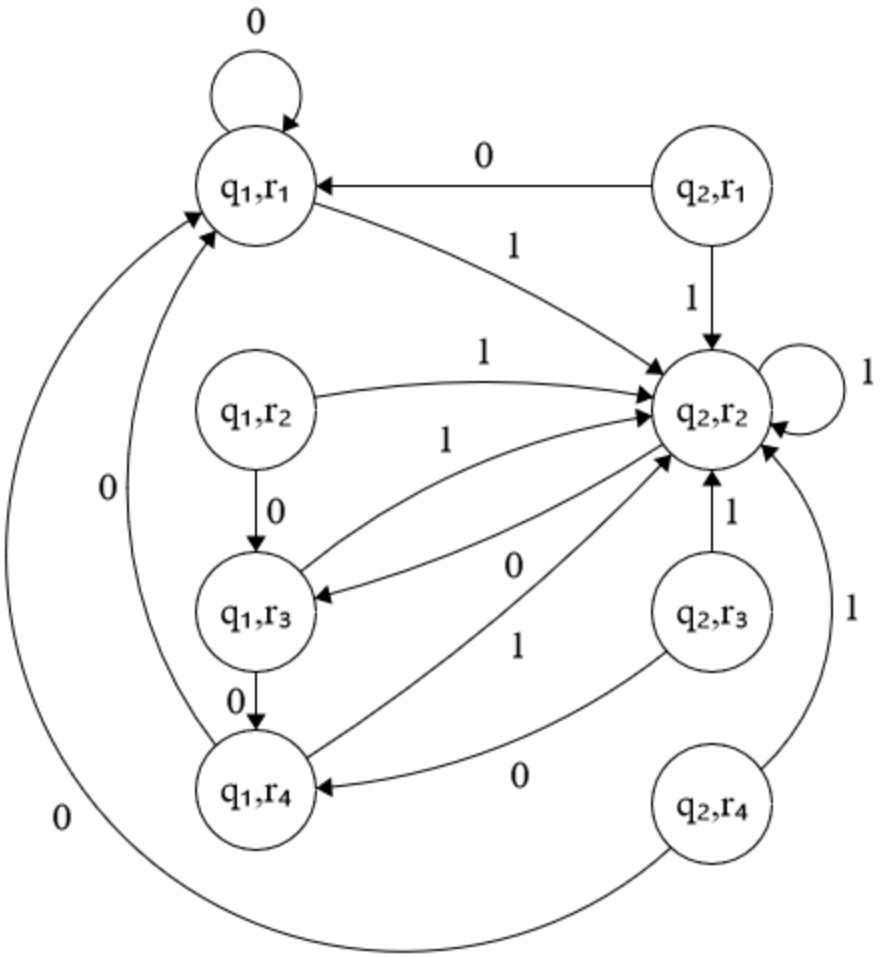
ต้องจำเยอะหน่อย ต้องจำอะไรบ้าง อะไรที่ต้องจำก็จะไปเป็น state

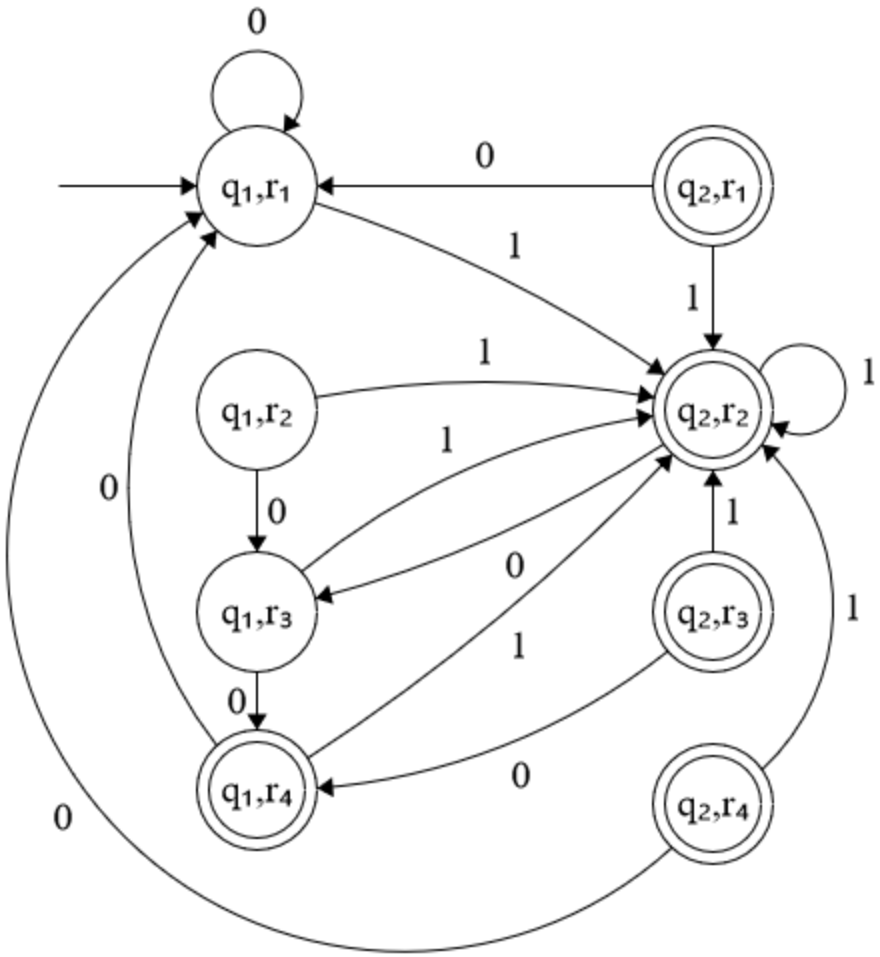
- ตัวแรกอยู่ที่ q_1 ตัวสองอาจจะอยู่ที่ r_1 หรือ r_2 หรือ r_3 หรือ r_4 ก็ได้
- ตัวแรกอยู่ที่ q_2 ตัวสองอาจจะอยู่ที่ r_1 หรือ r_2 หรือ r_3 หรือ r_4 ก็ได้

q_1,r_1	q_2,r_1
q_1,r_2	q_2,r_2
q_1,r_3	q_2,r_3
q_1,r_4	q_2,r_4


q_1,r_1
q_2,r_1
q_1,r_2
q_2,r_2
q_1,r_3
q_2,r_3
q_1,r_4
q_2,r_4


0
q1,r1
q2,r1
1
q1,r2
q2,r2
q1,r3
q2,r3
q1,r4
q2,r4


0
q1,r1
0
q2,r1
1
1
1
q1,r2
q2,r2
1
1
0
0
1
q1,r3
q2,r3
0
1
1
0
q1,r4
q2,r4
0
0


0
q1,r1
0
q2,r1
1
1
1
q1,r2
q2,r2
1
1
0
0
1
q1,r3
q2,r3
0
1
1
0
q1,r4
q2,r4
0
0

Machine M ที่ recognizing $A_1 \cup A_2$

กำหนดให้ Machine M_1 = $(Q_1, \Sigma, \delta_1, q_1, F_1)$ที่ recognize A_1

Machine M_2 = $(Q_2, \Sigma, \delta_2, q_2, F_2)$ที่ recognize A_2

Machine M =$(Q_1, \Sigma, \delta, q_0, F)$ที่

$Q = Q_1 \times Q_2$

Σ ยังคงเหมือนเดิม

δ นิยามโดย

$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

$q_0 = (q_1, q_2)$

$F = \{(r_1, r_2,) | r_1 \in F_1 or\ r_2 \in F_2\}$