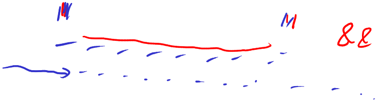
01204211 Discrete Mathematics
Lecture 9a: Finite automata[1]

Jittat Fakcharoenphol

August 29, 2024

[1]Based on lecture notes of *Models of Computation* course by Jeff Erickson.

Example: syntax highlighting



&& ((   ))

state machine.

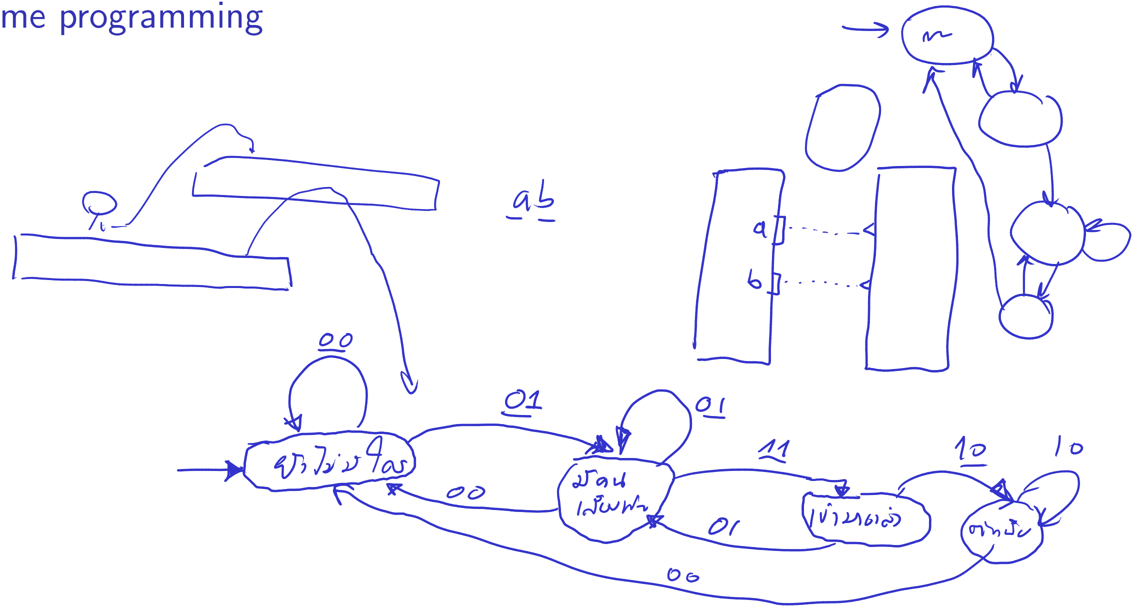
HTML tokenizer

<div>
 — — —
</div>

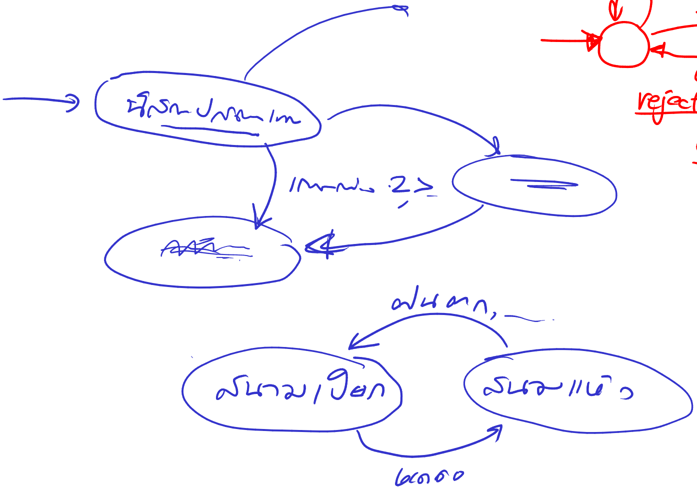
while ______ {
if(*p == - - -);
}
p++
}

state machine

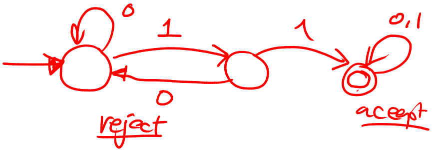
Game programming



State-transition graphs



Input string over {0,1}



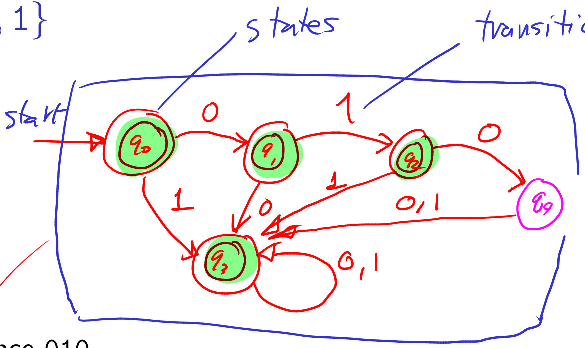
001010110101

"yes"

More examples over $\Sigma = \{0, 1\}$

All strings, except 010.

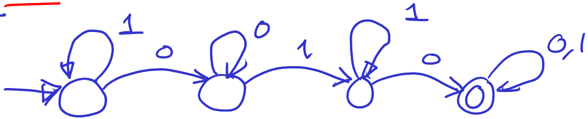
✓ 0100



δ	0	1
q_0	q_1	q_3
q_1	q_3	q_2
q_2	q_4	q_3
q_3	q_3	q_3
q_4	q_3	q_3

→ Strings containing the subsequence 010.

0011110



$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$Q \times \Sigma = \{(q_0, 0), (q_0, 1), (q_1, 0), \ldots$

Formal definitions

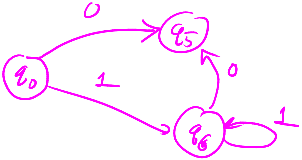
A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- the input alphabet Σ,

Formal definitions



A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- the input alphabet Σ,
- a finite set of states Q,

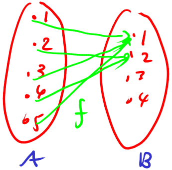
Formal definitions

$\delta: \underline{A} \to \underline{B}$

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- the input alphabet Σ,
- a finite set of states Q,
- a transition function δ

Formal definitions



$f : A \to B$

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

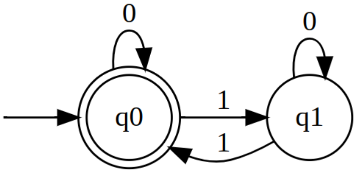
- the input alphabet Σ,
- a finite set of states Q,
- a transition function $\delta : \boxed{\overset{\text{state}}{Q} \times \overset{\text{symbol}}{\Sigma}} \longrightarrow Q$

Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- the input alphabet Σ,
- a finite set of states Q,
- a transition function $\delta : Q \times \Sigma \longrightarrow Q$
- a start state $s \in Q$, and
- a subset $A \subseteq Q$ of accepting states.

Example 1



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

$Q = \{q_0, q_1\}$

δ:

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

~~statat~~ start state $s = q_0$

accepting states $A = \{q_0\}$

$(\{0,1\}, \{\ \_\ \}, \delta, s, A)$

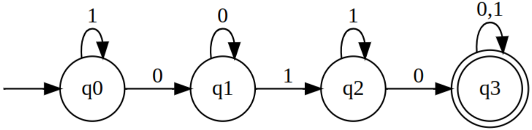
Example 2

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

$Q = \{$ ________

start state q_0

accepting states $\{q_3\}$



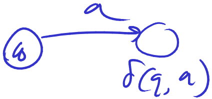
Moves

$\boxed{\text{อธิบาย}}$ พฤติกรรม ของ DFA

One step move: from state q with input symbol a, the machine changes its state to

$$\delta(q, a)$$

Moves

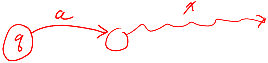


- **One step move**: from state q with input symbol a, the machine changes its state to $\delta(q, a)$.
- **Extension:** from state q with input string w, the machine changes its state to $\delta^*(q, w)$ defined as

Case: $w = \varepsilon$, $\quad \delta^*(q, w) = q$

Case: $w = a \cdot x$ where $a \in \Sigma$, x is a string

$$\delta^*(q, a \cdot x) = \delta^*\big(\delta(q, a), x\big)$$



Moves

One step move: from state q with input symbol a, the machine changes its state to $\delta(q, a)$.
Extension: from state q with input string w, the machine changes its state to $\delta^*(q, w)$ defined as

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon, \\ \delta^*(\delta(q, a), x) & \text{if } w = ax. \end{cases}$$

The signature of δ^* is $Q \times \Sigma^* \longrightarrow Q$.

Acceptance

For a finite-state machine with starting state s and accepting states A, it accepts string w iff

$$\delta^*(s, w) \in A$$

Acceptance

For a finite-state machine with starting state s and accepting states A, it accepts string w iff

$$\delta^*(s, w) \in A.$$

Multiple of 5

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

5, 10, 15, 25

101

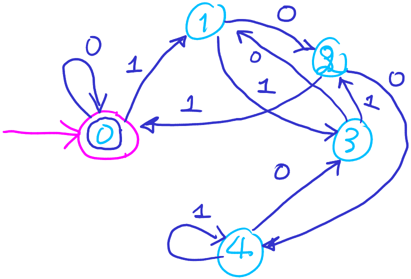
1010

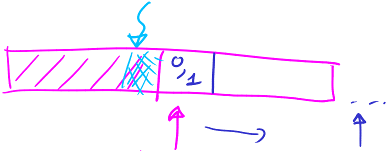
1111

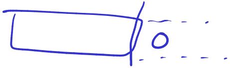
11001

$x \xrightarrow{1} 2x+1$

$x \xrightarrow{0} 2x$







$\delta(q, a)$

Multiple of 5

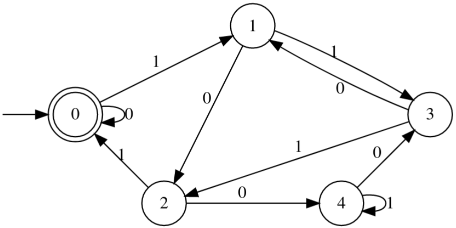
[0, 1, 1, 1, 0, 1, 0]

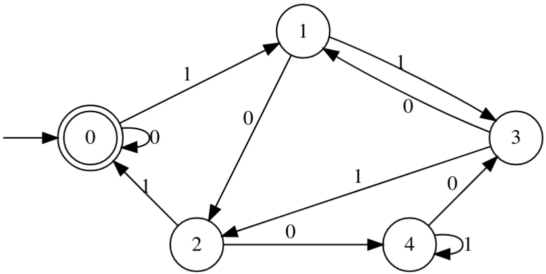
```
def multiple_of_5(w):
    r = 0
    for i in w:
        r = (2*r + w) % 5
    return r == 0
```

// $r = \delta(r, w)$

Multiple of 5

```
def multiple_of_5(w):
    r = 0
    for i in w:
        r = (2*r + w) % 5
    return r == 0
```




0
1
2
3
4
1
0
1
0
1
1
0
0
0
1

Digital design: Implementation

Digital design: Moore and Mealy machines

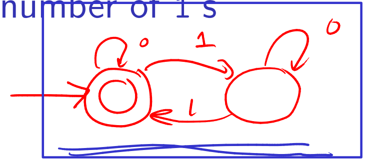
In the digital design class, you will encounter finite-state machines as well. The version we consider in this class is refered to as a **Moore machine**.
In practices, there is another variant of FSM called **Mealy machines**, whose outputs depend on input symbols as well.

Digital design: Moore and Mealy machines

In the digital design class, you will encounter finite-state machines as well. The version we consider in this class is refered to as a **Moore machine**.
In practices, there is another variant of FSM called **Mealy machines**, whose outputs depend on input symbols as well.
Formally, they differ in output function.

- Moore machine: $G : Q \longrightarrow [0,1]$
- Mealy machine: $G : Q \times \Sigma \longrightarrow [0,1]$

Example: even number of 1's



M_1

$(0^*10^*10^*)^*$

$L_1 = \{ w \mid w$ มี จน. 1 เป็นคู่ $\}$

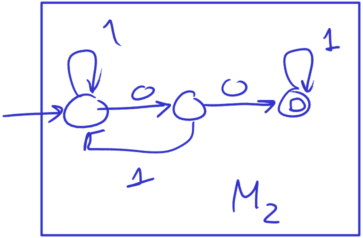
Example: strings containing 00 as a substring

$L_2 = \{ w \mid w$ has 00 as substring $\}$

$L_1 \cap L_2$

$L_1 \cup L_2$

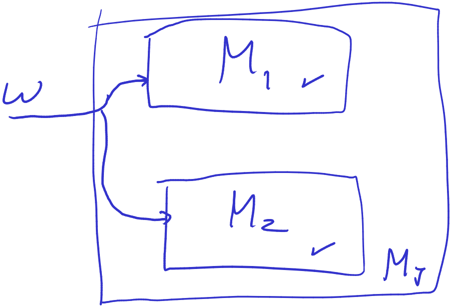
$(0+1)^* 00 (0+1)^*$

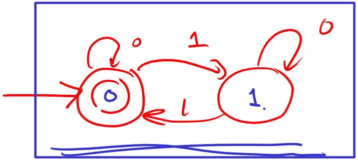


Combining DFAs

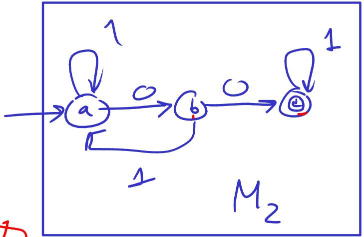
$\underline{0}11001$

What if we want to build a DFA that accepts strings with an even number of $\underline{1}$'s and containing $\underline{00}$ as a substring?





M_1

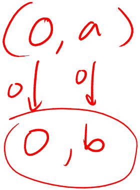


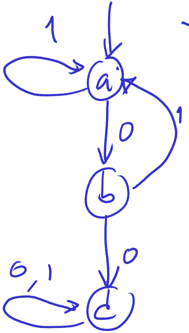
$$(q_0, r_0) \xrightarrow{0} (q_0, r_1) \xrightarrow{1} (q_1, r_0) \xrightarrow{1} (q_0, r_0) \xrightarrow{0} ($$

$\boxed{(q, r)}$

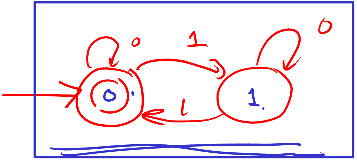
→ Product construction

$Q \times \Sigma$

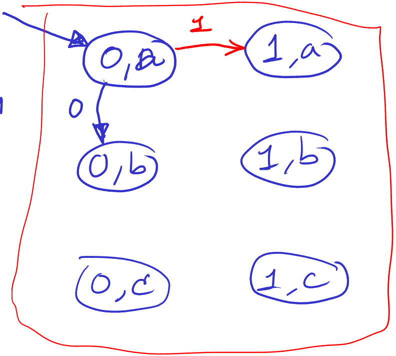




M_2



M_1



Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2, we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- Let $Q = Q_1 \times Q_2$,

$$\delta : Q \times \Sigma \to Q$$

$$\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow Q_1 \times Q_2$$

$$(q_i, r_j) \qquad a \qquad (\delta_1(q_i, a), \delta_2(q_j, a))$$

Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2, we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

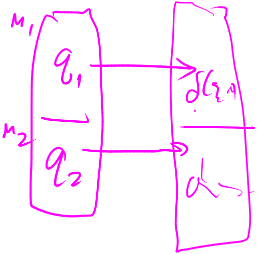
- Let $Q = Q_1 \times Q_2$,
- Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2, we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- Let $Q = Q_1 \times Q_2$,
- Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)),$$



Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2, we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- Let $Q = Q_1 \times Q_2$,
- Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)),$$

- Let $\underline{s} = \underline{(s_1, s_2)}$, and

Product construction (formally)

δ – one step

δ^*

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2, we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- Let $Q = Q_1 \times Q_2$,
- Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

$$\boxed{\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)),}$$

- Let $s = (s_1, s_2)$, and
- Let $A = A_1 \times A_2$.

Recall the definition of $\delta^*(q, w)$, i.e.,

Recall the definition of $\delta^*(q, w)$, i.e.,

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon, \\ \delta^*(\underbrace{\delta(q, a)}, x) & \text{if } w = ax \text{ where } a \in \Sigma \end{cases}$$

$\delta^*((q_1, q_2), w)$

Recall the definition of $\delta^*(q, w)$, i.e.,

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon, \\ \delta^*(\delta(q, a), x) & \text{if } w = ax \text{ where } a \in \Sigma \end{cases}$$

state ใน M_1 state ใน M_2

Lemma 1

$\delta^*((q_1, q_2), w) = (\delta_1^*(q_1, w), \delta_2^*(q_2, w))$ *for any string* w.

Proof.

We prove by induction. I.H.: Assume that $\delta^*((q_1, q_2), x) = (\delta_1^*(q_1, x), \delta_2^*(q_2, x))$, for any string x such that $|x| < |w|$.

Case: $w = \varepsilon$, $\delta^*((q_1, q_2), \varepsilon) = (q_1, q_2) = (\delta_1^*(q_1, \varepsilon), \delta_2^*(q_2, \varepsilon))$ ตามนิยาม

Case: $w = a \cdot x$ สำหรับ x ใด string

$$\delta^*((q_1, q_2), ax) = \delta^*(\delta((q_1, q_2), a), x) = \delta^*((\delta_1(q_1, a), \delta_2(q_2, a)), x)$$

จาก I.H

$$= (\delta_1^*(\delta_1(q_1, a), x), \delta_2^*(\delta_2(q_2, a), x)) = (\delta_1^*(q_1, ax), \delta_2^*(q_2, ax)) = (\delta_1^*(q_1, w), \delta_2^*(q_2, w))$$

Correctness

M ถูกต้อง $\underline{M}$ accepts string w iff M_1 & M_2 accept w

$L_1 \cap L_2$.

From the previous lemma, we have that

$$\delta^*(s,w) = \delta^*((s_1,s_2),w)$$

$$= (\delta_1^*(s_1,w), \delta_2^*(s_2,w))$$

Correctness

From the previous lemma, we have that

$$\begin{aligned}\delta^*(s, w) &= \delta^*((s_1, s_2), w) \\ &= (\delta_1^*(s_1, w), \delta_2^*(s_2, w))\end{aligned}$$

Correctness

From the previous lemma, we have that

$$\begin{aligned}\delta^*(s,w) &= \delta^*((s_1,s_2),w)\\ &= (\delta_1^*(s_1,w),\delta_2^*(s_2,w))\end{aligned}$$

Thus, for an input w, M would reach the state $(\delta_1^*(s_1,w),\delta_2^*(s_2,w))$; it accepts w when

$$(\delta_1^*(s_1,w),\delta_2^*(s_2,w)) \in \boxed{A_1 \times A_2.}$$

This implies that M accepts w when $\delta_1^*(s_1,w) \in A_1$ and $\delta_2^*(s_2,w) \in A_2$, i.e., M accepts w iff M_1 and M_2 accept w.
Finally, we conclude that M accepts strings from language $L_1 \cap L_2$.

Language of a DFA

$L(M)$

For a DFA M, let $L(M)$ be the set of all strings that M accepts. More formally, for $M = (\Sigma, Q, \delta, s, A)$,

$$L(M) = \{w \in \Sigma^* \mid \delta^*(s, w) \in A\}.$$

We refer to $L(M)$ as the language of M.

Closure

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2, we have that

- *there is a DFA M that accepts $L_1 \cap L_2$,*

Closure

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2, we have that

- *there is a DFA M that accepts $L_1 \cap L_2$,*
- *there is a DFA $\underline{M}$ that accepts $L_1 \cup L_2$,*

$A = \{(q_1, q_2) \mid$

$q_1 \in A_1$ или

$q_2 \in A_2\}$

Closure

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2, we have that

- *there is a DFA M that accepts $L_1 \cap L_2$,*
- *there is a DFA M that accepts $L_1 \cup L_2$,*
- *there is a DFA M that accepts $L_1 \setminus L_2$,*

Closure

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2, we have that

- *there is a DFA M that accepts $L_1 \cap L_2$,*
- *there is a DFA M that accepts $L_1 \cup L_2$,*
- *there is a DFA M that accepts $L_1 \setminus L_2$,*
- *there is a DFA M that accepts $\Sigma^* \setminus L_1$,*

Automatic languages[2]

Definition (for now)

A language L is **"automatic"** if there is a DFA M such that $L(M) = L$.

[2] Taken directly from Erickson's lecture notes

Automatic languages[2]

Definition (for now)

A language L is **"automatic"** if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ, then

- $L_1 \cap L_2$,

[2] Taken directly from Erickson's lecture notes

Automatic languages[2]

Definition (for now)

A language L is **"automatic"** if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ, then

- $L_1 \cap L_2$,
- $L_1 \cup L_2$,

[2] Taken directly from Erickson's lecture notes

Automatic languages[2]

Definition (for now)

A language L is **"automatic"** if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ, then

- $L_1 \cap L_2$,
- $L_1 \cup L_2$,
- $L_1 \setminus L_2$, *and*

[2] Taken directly from Erickson's lecture notes

Automatic languages[2]

Definition (for now)

A language L is **"automatic"** if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 *and* L_2 *are automatic languages over alphabet* Σ, *then*

- $L_1 \cap L_2$,
- $L_1 \cup L_2$,
- $L_1 \setminus L_2$, *and*
- $\Sigma^* \setminus L_1$,

are also automatic.

คุณสมบัติปิด

[2] Taken directly from Erickson's lecture notes

Automatic languages[2]

Definition (for now)

A language L is **"automatic"** if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ, then

- $L_1 \cap L_2$,
- $L_1 \cup L_2$,
- $L_1 \setminus L_2$, *and*
- $\Sigma^* \setminus L_1$,

are also automatic.

The set of automatic languages is closed under these boolean operations.

[2] Taken directly from Erickson's lecture notes