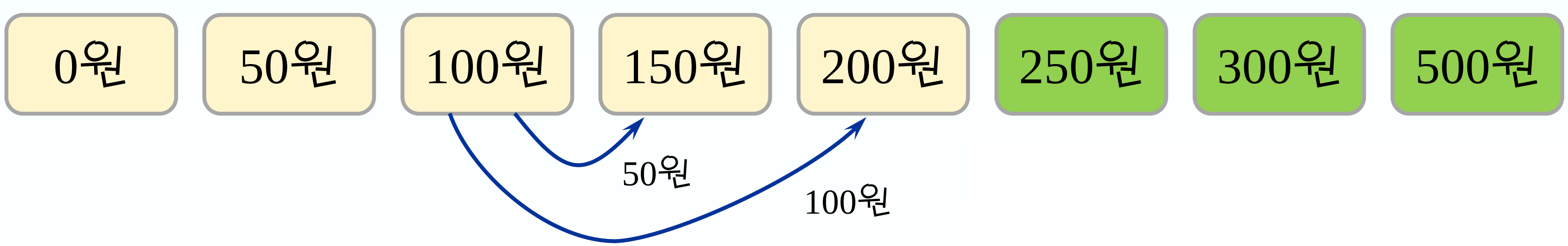
Finite Automata

부산대학교
PUSAN NATIONAL UNIVERSITY

Finite State Machine

- Finite State Machines (FSM)
  - **It consists of a finite number of internal states and transitions among them.**
  - **It remembers certain information when it is in a particular state.**

(예) 250원짜리 커피 자판기 (50, 100, 500원 동전만 사용)
동전 투입 과정에서 기억해야 할 정보는?



An Example

- {0,1} 알파벳으로 구성된 스트링 중에서, 끝에서 두 번째 문자가 1인 것들의 집합 (언어)

{ 10, 11, 010, 011, 110, 111, … }

Finite Automata

(예) 0 1 1 1 0

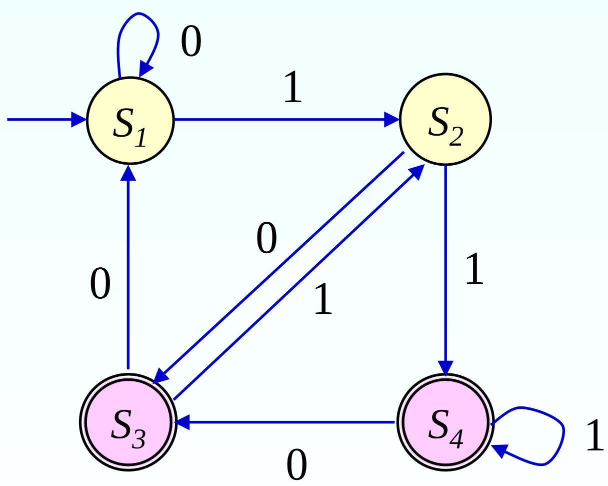
$S_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_4 \rightarrow S_3$

λ + 0 + (0+1)*00 상태

1 + (0+1)*01 상태

(0+1)*11 상태

(0+1)*10 상태



DFA

Deterministic Finite Automata

부산대학교
PUSAN NATIONAL UNIVERSITY

DFA

- Definition

  A deterministic finite automata (DFA) M is specified by a quintuple

$$M = (Q, \sum, \delta, q_0, F)$$

- Q is an alphabet of state symbols ;
- $\sum$ is an alphabet of input symbols ;
- $\delta : Q \times \sum \rightarrow Q$ is a transition function ;
- $q_0 \in Q$ is the start state; and
- $F \subseteq Q$ is a set of final states.

single, double, triple, quadruple, quintuple, sextuple, …

State Diagram

- State Diagram of $M = (Q, \Sigma, \delta, q_0, F)$

  State symbols in Q ; vertices (원)

  Input symbols in Σ ; labels

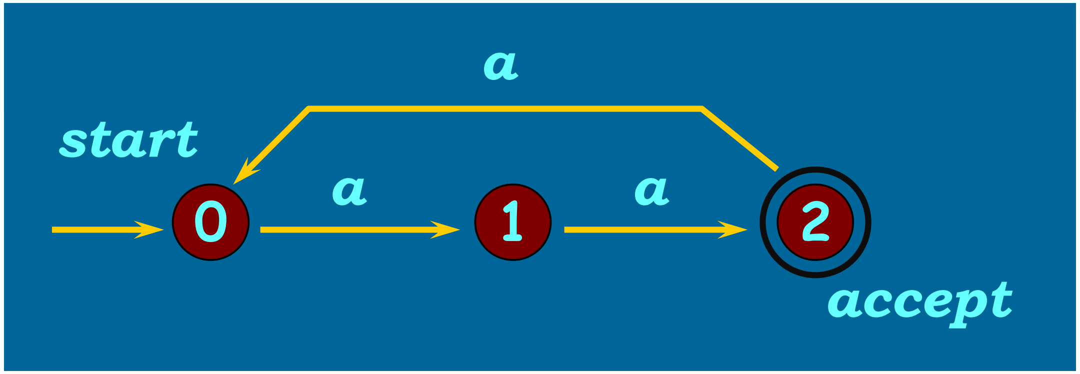
  $\delta : Q \times \Sigma \rightarrow Q$: edges with labels (화살표)

  $q_0 \in Q$: start로 명시된 vertex (구별되는 화살표)

  $F \subseteq Q$: 이중 원

A DFA Example

- $M = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{ 0, 1, 2 \}$, $\Sigma = \{ a \}$
  - $\delta : Q \times \Sigma \rightarrow Q$
    - $\delta(0, a) = 1$, $\delta(1, a) = 2$, $\delta(2, a) = 0$
  - $q_0 = 0$, $F = \{ 2 \}$



Another DFA Example

- $M = (Q, \sum, \delta, q_0, F)$

  $Q = \{ S_1, S_2, S_3, S_4 \}$

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

  $\delta : Q \times \sum \rightarrow Q$

  $\delta(S_1,0) = S_1 , \ \delta(S_1,1) = S_2$

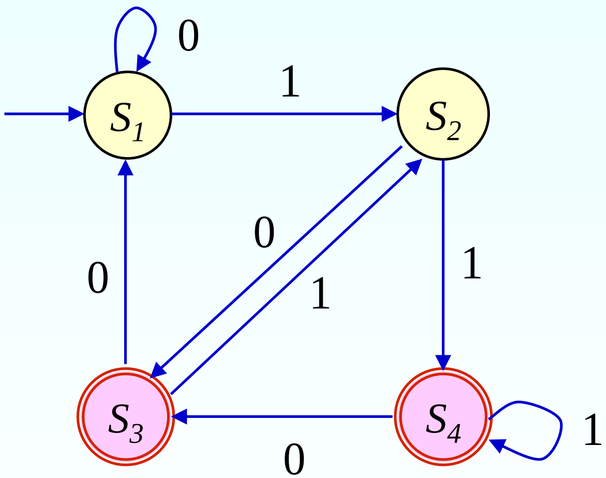
  $\delta(S_2,0) = S_3 , \ \delta(S_2,1) = S_4$

  $\delta(S_3,0) = S_1 , \ \delta(S_3,1) = S_2$

  $\delta(S_4,0) = S_3 , \ \delta(S_4,1) = S_4$

  $q_0 = S_1$

  $F = \{ S_3, S_4 \}$

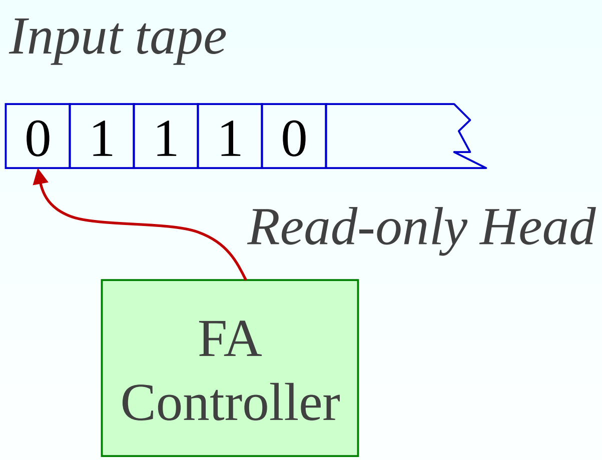


Machine-Oriented Viewpoint

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

  We view it as a machine (a primitive computer)

  - It has an input tape of cells, a read-only head, and a FA controller



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

- No memory
- Read-only head
- Move the head to the right
- Initially at the start state & at the leftmost position
- δ (current state, tape symbol) → (next state)

DFA Configuration

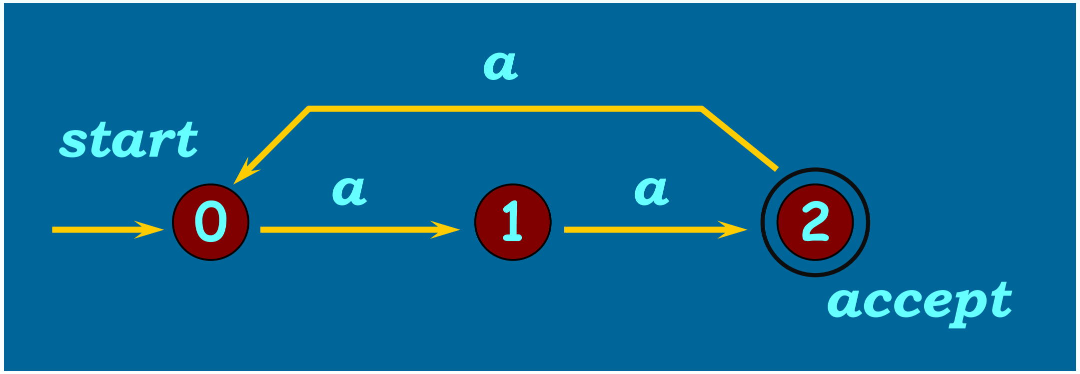
- Definition

  Let $M = (Q, \Sigma, \delta, S, F)$ be a DFA.

  상황

  We say that a word in $Q\Sigma^*$ is a configuration of M.

  The word presents the current state of M and the remaining unread input of M.



0 aaa

1 aa

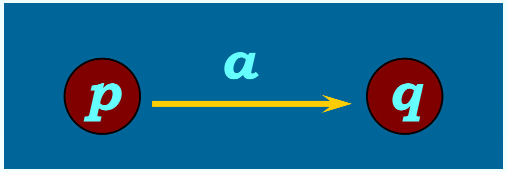
2 a

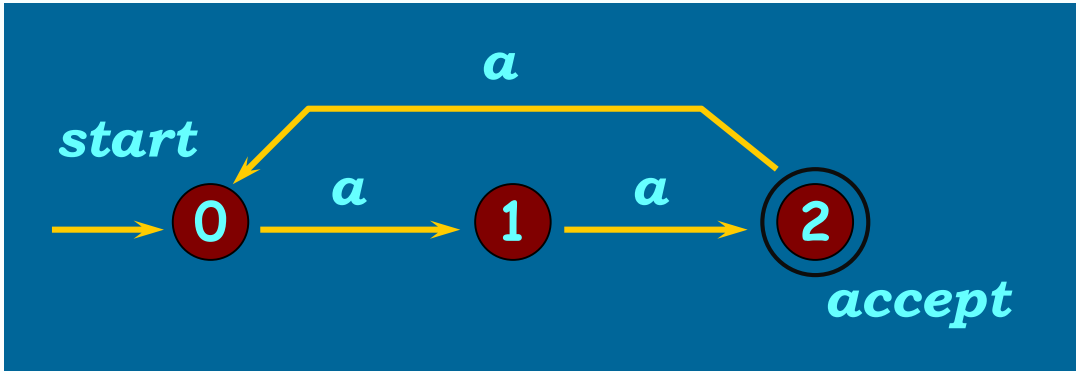
0 λ

Configuration Sequence

- Definition

  Let px and qy are two configurations of a DFA M.
  We write $px \vdash qy$, if $x = ay$ for some $a \in \Sigma$ and $\delta(p,a) = q$.





$1aa \vdash 2a$

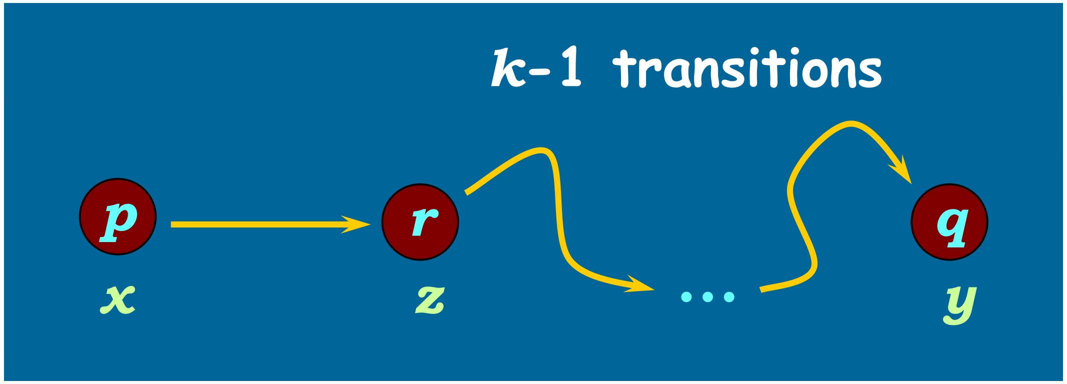
$2a \vdash 0\lambda$

Configuration Sequence

- **Definition**

  For $k \geq 1$, we write $px \vdash^k qy$ (k-steps of M on px), if

  1) $k = 1$ and $px \vdash qy$ or
  2) $k > 1$ and there exists a configuration rz such that $px \vdash rz$ and $rz \vdash^{k-1} qy$.



Configuration Sequence

For the binary relation $\vdash$ on $Q\Sigma^*$

$px \vdash qy$ i.e. $(px, qy) \in \vdash$

$\vdash^+$: transitive closure $t(\vdash)$ of $\vdash$

$px \vdash^+ qy$ and $qy \vdash^+ rz \Rightarrow px \vdash^+ rz$

$(\equiv \vdash^k, k \geq 1)$

$\vdash^*$: reflexive transitive closure $rt(\vdash)$ of $\vdash$

$px \vdash^* qy$ and $qy \vdash^* rz \Rightarrow px \vdash^* rz$

$(\equiv \vdash^k, k \geq 0)$

Review of transitive closure

If $\mathcal{R}$ is a relation on a set A then the transitive (symmetric, reflexive) *closure* of $\mathcal{R}$ is a relation $\mathcal{R}'$ such that

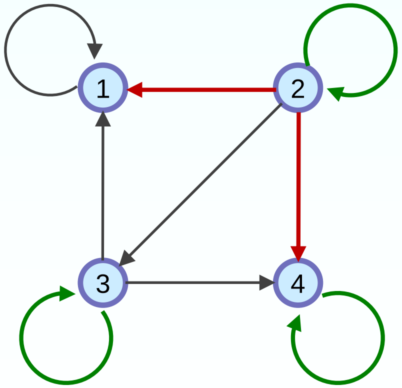
1. $\mathcal{R}'$ is transitive (symmetric, reflexive)
2. $\mathcal{R} \subseteq \mathcal{R}'$
3. If $\mathcal{R}''$ is another transitive (symmetric, reflexive) relation and $\mathcal{R} \subseteq \mathcal{R}''$, then $\mathcal{R}' \subseteq \mathcal{R}''$.

(Examples)

$\mathcal{R} = \{(1,1),(2,3),(3,4),(3,1)\}$

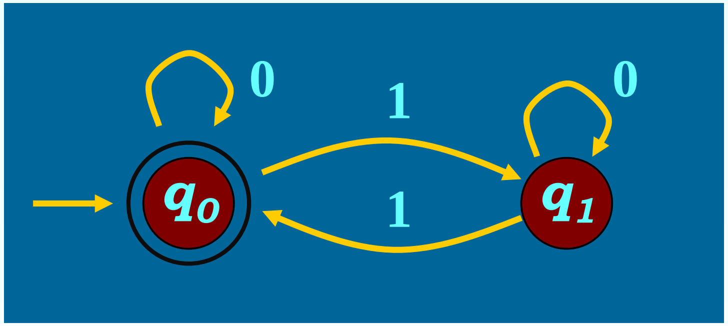
$t(\mathcal{R}) = \{(1,1),(2,3),(3,4),(3,1),(2,1),(2,4)\}$

$rt(\mathcal{R}) = \{(1,1),(2,3),(3,4),(3,1),(2,1),(2,4)$
$,(2,2),(3,3),(4,4)\}$



Configuration Sequence

We say that the sequence of configuration given by $px \vdash^* qy$ is a configuration sequence.



$q_0\ 1010 \vdash^+ q_1\ 010$

$q_0\ 1010 \vdash^+ q_1\ 10$

$q_0\ 1010 \vdash^+ q_0\ \lambda$

$q_0\ 1010 \vdash^* q_1\ 010$

$q_1\ 010 \vdash^* q_1\ 010$

$q_1\ 010 \vdash^* q_0\ \lambda$

$q_0\ 1010 \vdash^* q_0\ \lambda$

Accepted Language $L(M)$

- Definition

Let $M = (Q, \Sigma, \delta, S, F)$ be a DFA.

We say that a string x in Σ^* is accepted by M, if $Sx \vdash^* f\lambda$, for some f in F.

We say that $Sx \vdash^* f\lambda$ is an accepting configuration sequence.

The set of strings accepted by M, called the language accepted, defined, or recognized by M is denoted by $L(M)$ and is defined as

$L(M) = \{ x \mid x \in \Sigma^* \text{ and } Sx \vdash^* f\lambda, \text{ for some } f \text{ in } F \}$

DFA Language

- DFA Language

  We say that $L \subseteq \Sigma^*$ is a DFA language if there is a DFA M, with $L = L(M)$.

- Equivalence of DFAs

  Let M_1 and M_2 be two DFAs.

  If $L(M_1) = L(M_2)$, we say that M_1, M_2 are equivalent.

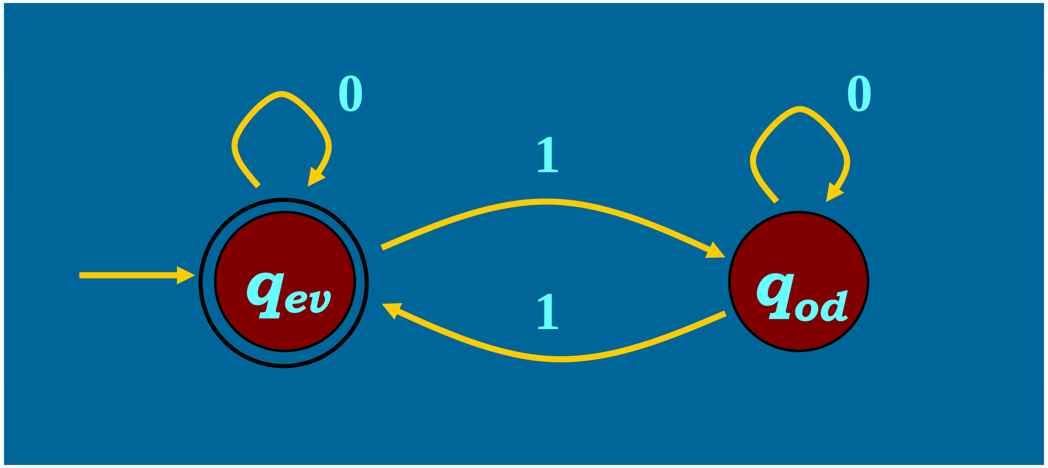
How to construct DFAs

- DFA는 다음 두 단계에 의해서 만든다
  1. 현재 까지 읽은 부분에서 어떤 정보를 기억해야 하는지를 정하고 이 정보를 state로 표시한다.
  2. 새로운 input symbol을 읽었을 때 기억해야 하는 정보가 어떻게 바뀌는지를 보고 transition function 을 만든다.

- 주의할 점
  - 중요한 것은 기억해야 하는 정보를 정하는 것이다.
  - 만들고자 하는 DFA의 기능을 분석하여 기억해야 하는 정보를 정리하고 state로 매핑한다.

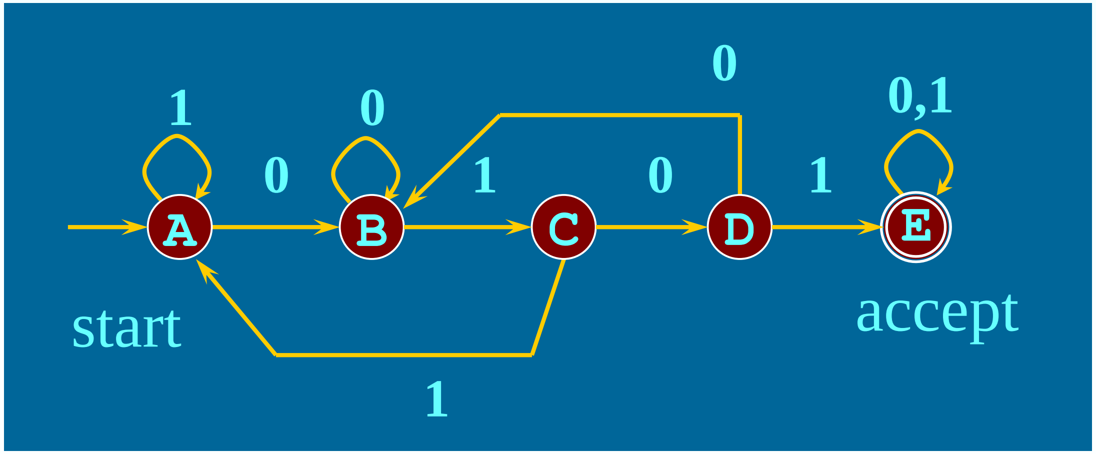
Example (1)

- 짝수 개의 1을 갖는 string을 accept하는 DFA를 구하라.
  - 현재까지 읽은 substring에서 1의 개수에 대한 짝수 또는 홀수 여부를 상태로 기억해야 함.



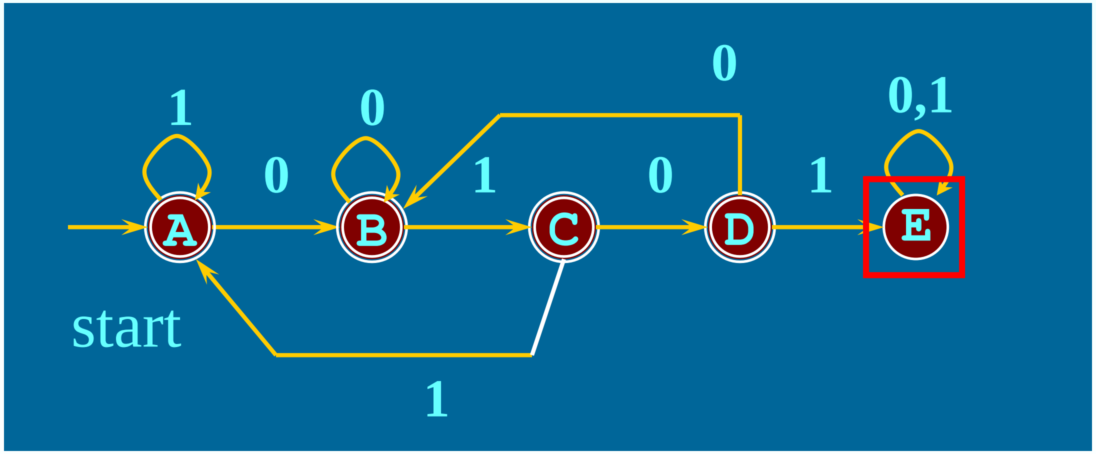
Example (2)

- 0101을 substring으로 가지는 string을 accept 하는 DFA를 구하라.
  - 현재까지 read한 부분이 0101의 prefix 중에서 어떤 것에 해당하는 지를 기억해야 함.
  - 따라서 0101의 prefix 각각에 해당하는 state를 만들면 됨. (λ, 0, 01, 010, 0101)



Example (3)

- 0101을 substring으로 갖지 않는 string을 accept하는 DFA를 구하라.
  - 앞 예제의 final state와 그 외 상태들의 기능을 맞바꾸면 됨.



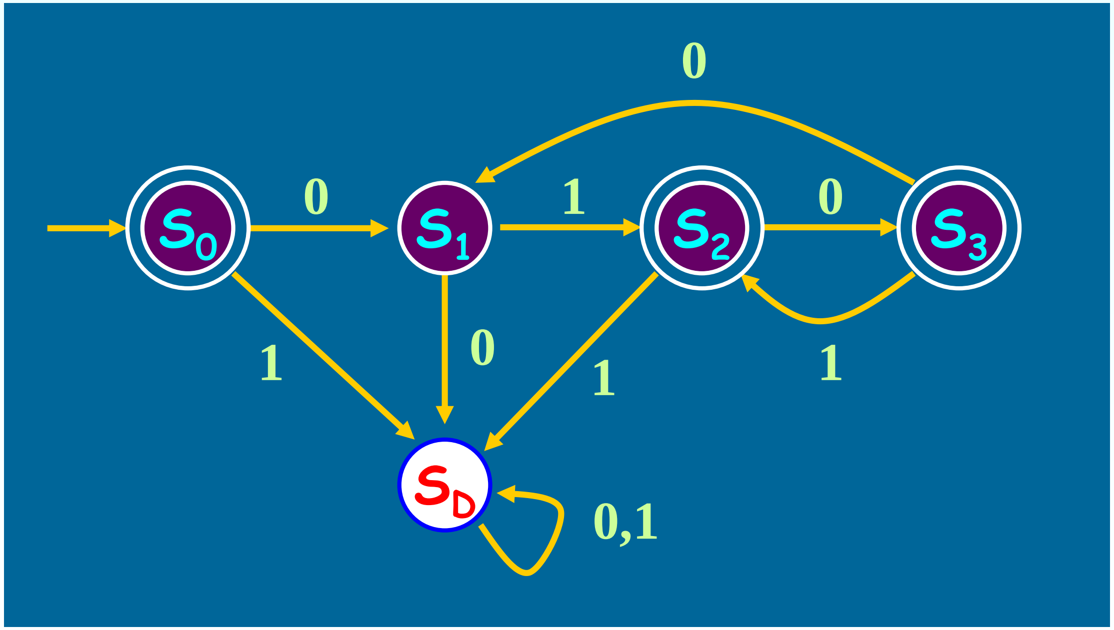
Dead State

한번 들어가면 빠져 나오지 못하는 상태

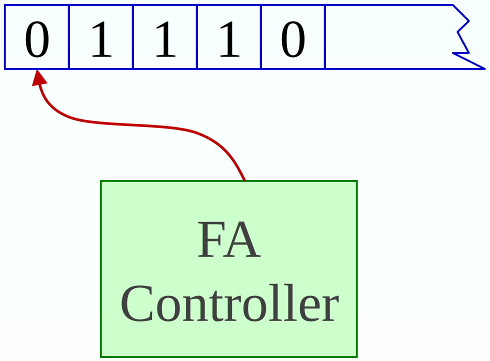
Another Example of Dead States

- (010+01)* string을 accept하는 DFA?

Prefix : λ, 0, 01, 010



Implementation of FA Controller



부산대학교
PUSAN NATIONAL UNIVERSITY

A Mealy-type DFA

- Def. A Mealy-type FA

$$M = (Q, \Sigma, \Gamma, \delta, \gamma)$$

Q : a set of finite states

Σ : an input alphabet

Γ : an output alphabet

δ : next state function $Q \times \Sigma \rightarrow Q$

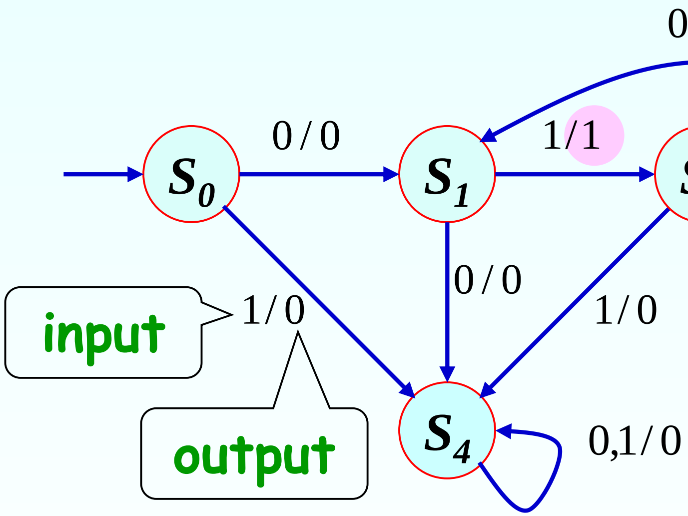
γ : an output function $Q \times \Sigma \rightarrow \Gamma$ Output(state, input)

Output(state)

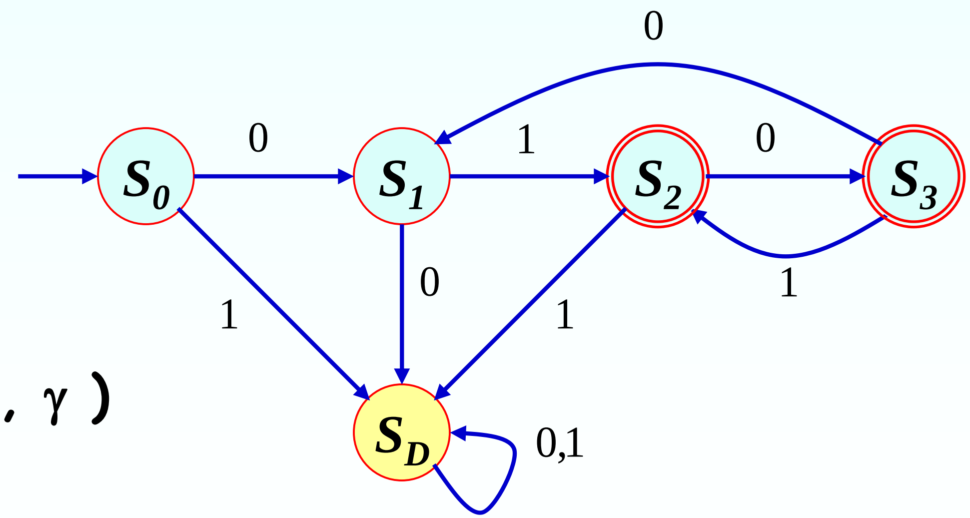
(note) DFA $M = (Q , \Sigma , \delta , q_0 , F)$ $\leftarrow$ Moore-type

An Example of the M-FA

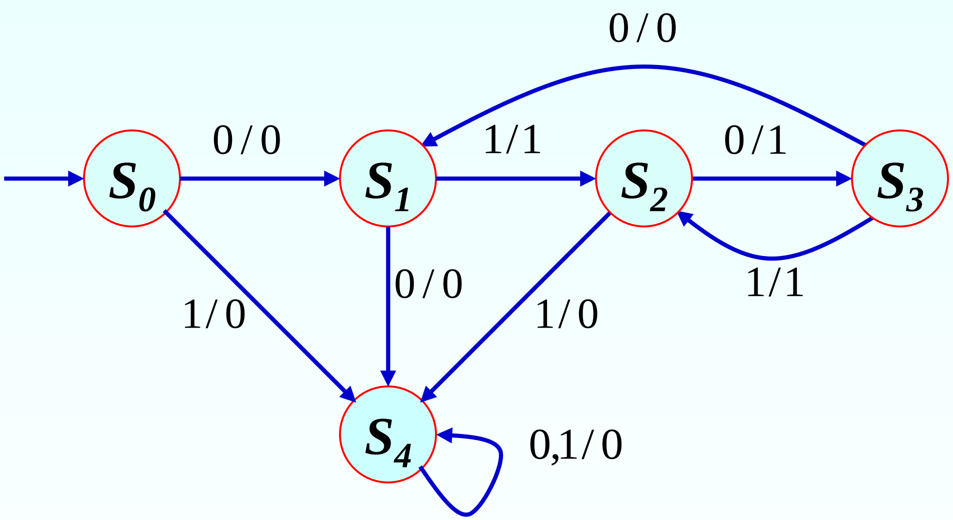
- (010+01)$^+$ string을 accept하는 DFA?



M = ({S_0,…,S_4}, {0,1}, {0,1}, δ, γ)



❑ $(010+01)^+$ string을 accept하는 DFA?



$M = (\{S_0,\dots,S_4\}, \{0,1\}, \{0,1\}, \delta, \gamma)$

	δ		γ	
	0	1	0	1
S_0	S_1	S_4	0	0
S_1	S_4	S_2	0	1
S_2	S_3	S_4	1	0
S_3	S_1	S_2	0	1
S_4	S_4	S_4	0	0

State (Transition) Table

State Assignment & F/F Excitation Table

Current State Next State

C.S.	I	N.S.	O
A B C	x	A B C	y
0 0 0	0	0 0 1	0
0 0 0	1	1 0 0	0
0 0 1	0	1 0 0	0
0 0 1	1	0 1 0	1
0 1 0	0	0 1 1	1
0 1 0	1	1 0 0	0
0 1 1	0	0 0 1	0
0 1 1	1	0 1 0	1
1 0 0	0	1 0 0	0
1 0 0	1	1 0 0	0

Flip-Flop

Q	Q'	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Q	Q'	D
0	0	0
0	1	1
1	0	0
1	1	1

	δ		γ	
	0	1	0	1
S_0	S_1	S_4	0	0
S_1	S_4	S_2	0	1
S_2	S_3	S_4	1	0
S_3	S_1	S_2	0	1
S_4	S_4	S_4	0	0

ABC

$S_0 : 000$ $S_1 : 001$ $S_2 : 010$

$S_3 : 011$ $S_4 : 100$

Input Equations

C.S.	I	N.S.	O
A B C	X	A B C	Y
0 0 0	0	0 0 1	0
0 0 0	1	1 0 0	0
0 0 1	0	1 0 0	0
0 0 1	1	0 1 0	1
0 1 0	0	0 1 1	1
0 1 0	1	1 0 0	0
0 1 1	0	0 0 1	0
0 1 1	1	0 1 0	1
1 0 0	0	1 0 0	0
1 0 0	1	1 0 0	0

AB \ CX			
	1		1
	1		
x	x	x	x
1	1	x	x

$D_A = A + C'X + B'CX'$

		1	
1		1	
x	x	x	x
		x	x

$D_B = CX + BC'X'$

1			
1			1
x	x	x	x
		x	x

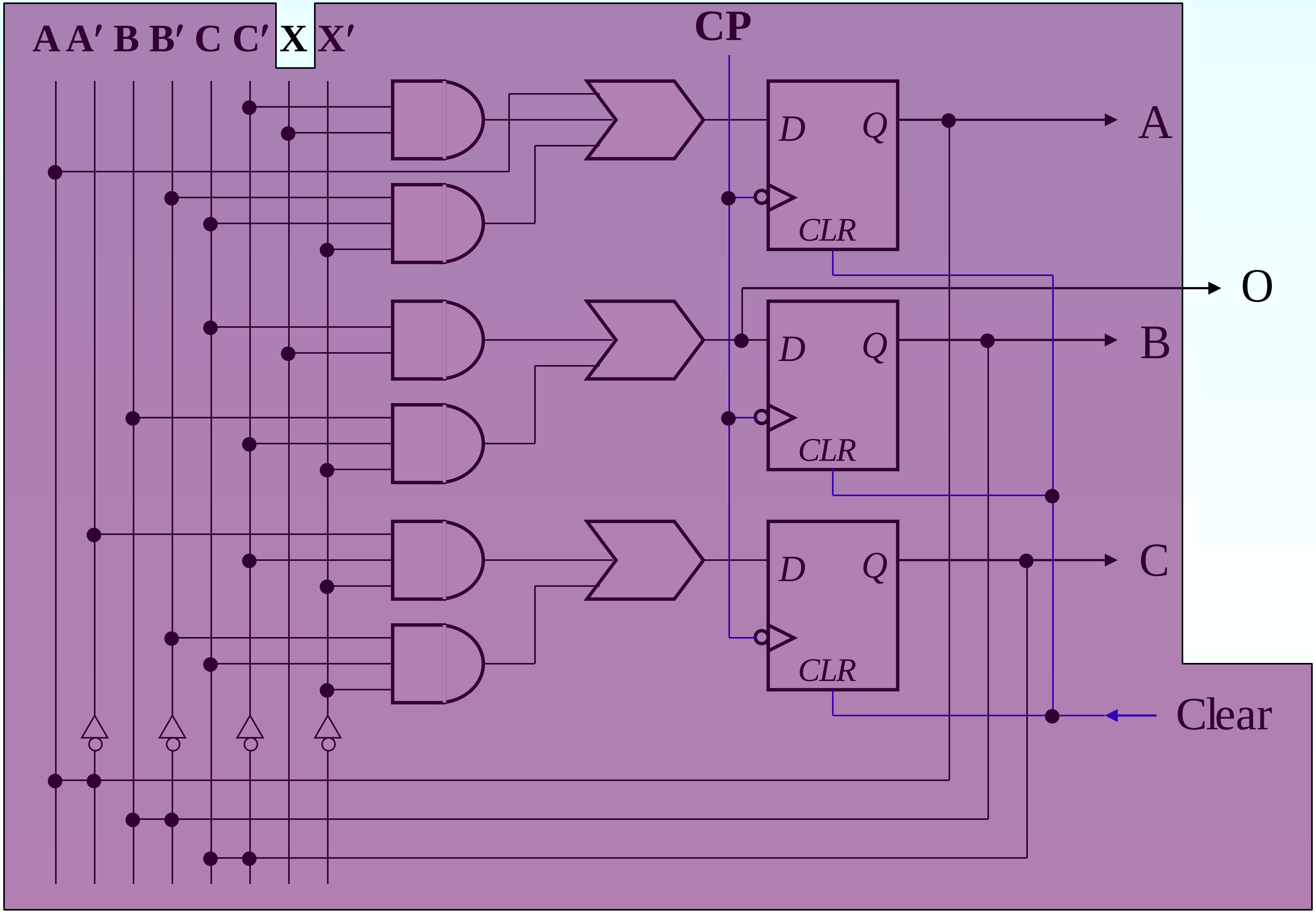
$D_C = A'C'X' + B'CX'$

		1	
1		1	
x	x	x	x
		x	x

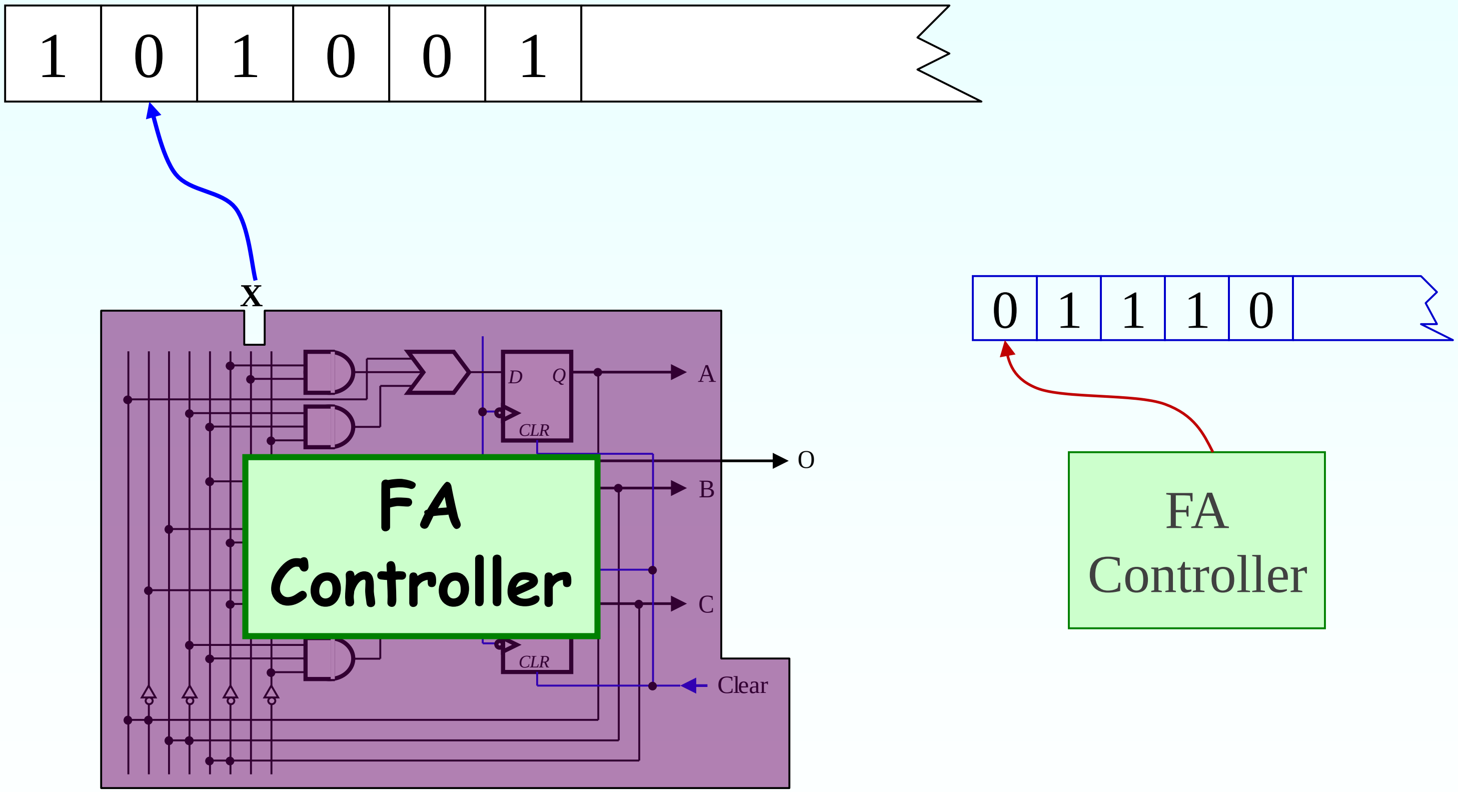
$O = D_B$

continued

$D_A = A + C'X + B'CX'$
$D_B = CX + BC'X'$
$D_C = A'C'X' + B'CX'$
$O = D_B$



continued



1
0
1
0
0
1
X
A
O
B
C
Clear
D
Q
CLR
CLR
FA
Controller
0
1
1
1
0
FA
Controller