계산이론

2022년 1학기

이은주

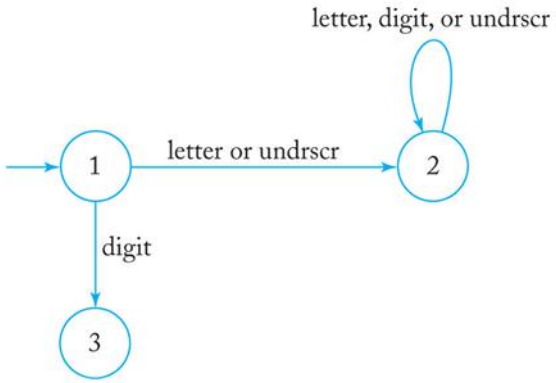
2장 유한 오토마타
결정적 유한 인식기
정규언어
비결정적 유한 인식기

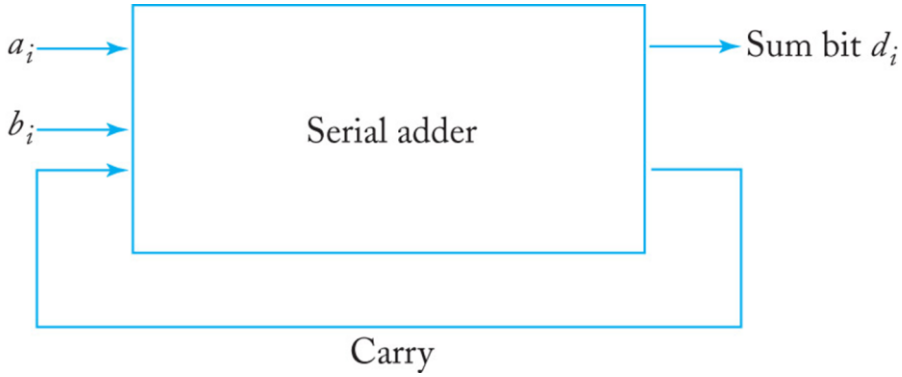
요약

- 유한 인식기
  - 유한 개수의 상태 : 유한
  - 문자열 승인(accept) 또는 거부(reject) : 인식기
  - 패턴인식
- 결정적 유한 인식기(dfa : deterministic finite accepter)
  - 오토마타는 오직 하나의 선택만 가능 : 결정적
  - 정규 언어(regular language) 정의
  - 오토마타를 언어와 연결
- 비결정적 오토마타(nfa : nondeterministic automata)
- 두 인식기의 동치 : 같은 언어 승인

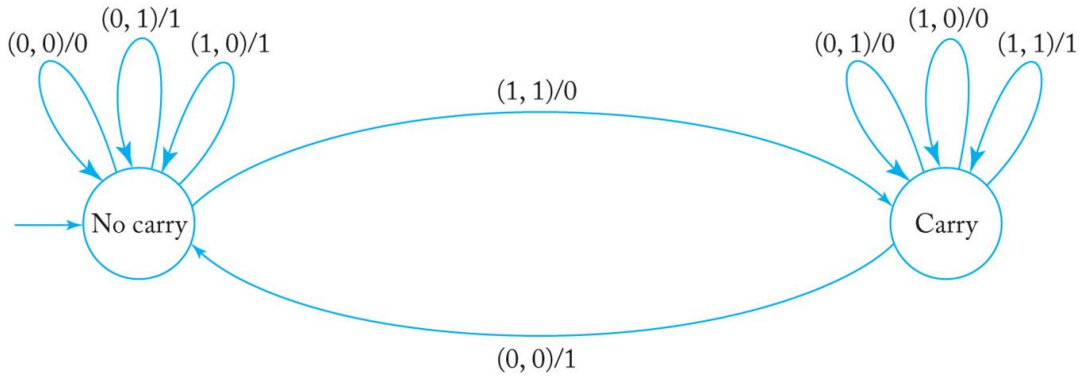
결정적 유한 인식기

- 연산 과정 : 결정적으로 진행





a_i \ b_i	0	1
0	0 No carry	1 No carry
1	1 No carry	0 Carry



결정적 유한 인식기(dfa : deterministic finite accapter)

- [정의] 결정적 유한 인식기, 결정적 유한 오토마타(dfa : deterministic dinite automata)

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

  (1) Q : 내부 상태(internal state)들의 유한집합

  (2) $\sum$: 입력기호(input symbol)들의 유한 집합, 입력 알파벳(input alphabet)

  (3) $\delta : Q \times \sum \rightarrow Q$: 전체 함수(total function), 전이함수(transition function), 상태 전이 함수(state transition function)

  (4) $q_0 \in Q$: 초기 상태(initial state)

  (5) $F \subseteq Q$: 승인 상태(final state), 최종 상태의 집합

$$\delta(q_0, a) = q_1$$

결정적 유한 인식기(dfa : deterministic finite accapter)

- [정의] **상태 전이도**(State Transition Graph(Diagram))는 다음과 같은 그래프(graph)
  (1) 노드(node) : 상태를 나타냄
  (2) 전이함수 $\delta(p,a) = q$ 에 대해 ⓟ$\rightarrow^{a}$ⓠ 가 존재한다.
  (3) 초기 노드 : →ⓢ
  (4) 최종 노드 : 이중 원(circle)

결정적 유한 인식기(dfa : deterministic finite accapter)

- [정의] **상태 전이표**(State Transition Table)

dfa M	전이함수 δ	input	
		0	1
state	초기 q_0	q_0	q_1
	최종 q_1	q_0	q_2
	q_2	q_0	q_1

dfa δ	0	1
초기 q_0	q_0	q_1
최종 q_1	q_0	q_2
q_2	q_0	q_1

결정적 유한 인식기(dfa : deterministic finite accapter)

- **예제 2.1(교재 46page) dfa** M = ($\{q_0, q_1, q_2\}$, {0, 1}, δ, q_0, $\{q_1\}$)

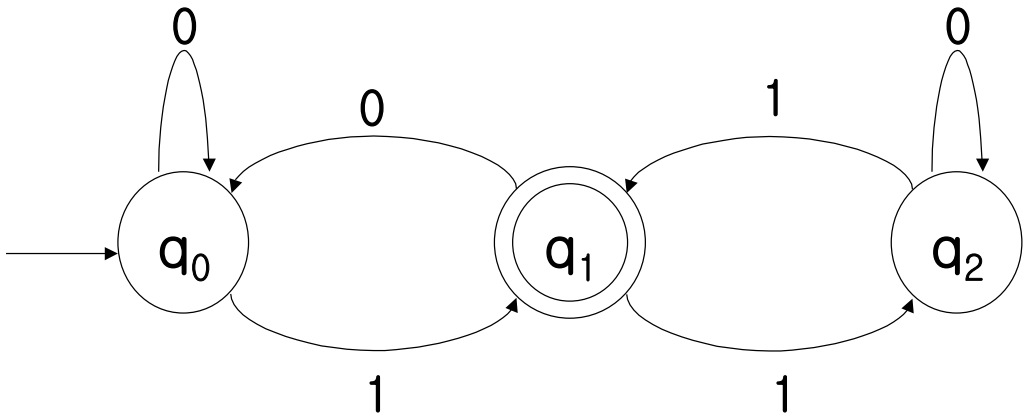
  δ :

  $\delta(q_0, 0) = q_0$ $\delta(q_0, 1) = q_1$

  $\delta(q_1, 0) = q_0$ $\delta(q_1, 1) = q_2$

  $\delta(q_2, 0) = q_2$ $\delta(q_2, 1) = q_1$

전이도



전이표

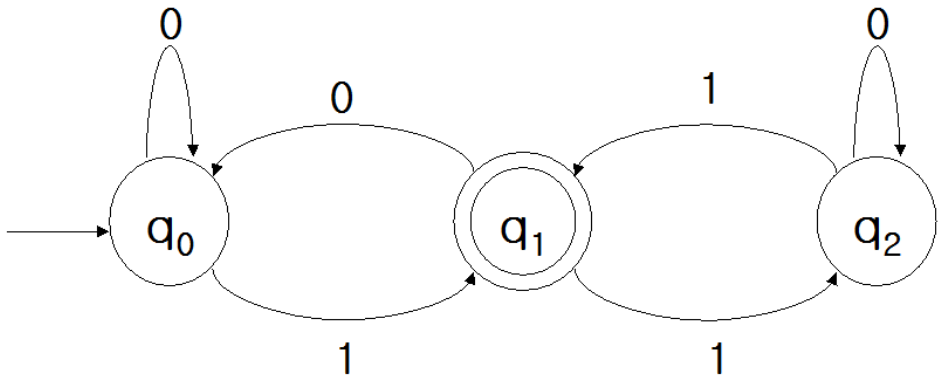
DFA δ	0	1
초기 q_0	q_0	q_1
최종 q_1	q_0	q_2
q_2	q_2	q_1

Accept : the string 111
Reject : the string 110

확장 전이 함수(extended transition function)

- $\delta^* : Q \times \Sigma^* \rightarrow Q*$

- $\delta^*(q_0, 1001) = q_1$
- $\delta^*(q_1, 000) = q_0$



결정적 유한 인식기(dfa : deterministic finite accapter)

- **(q, v) : 형상**(**configuration**, description of automata)
  q : 현재 상태
  v : 아직 읽지 않은 입력스트링

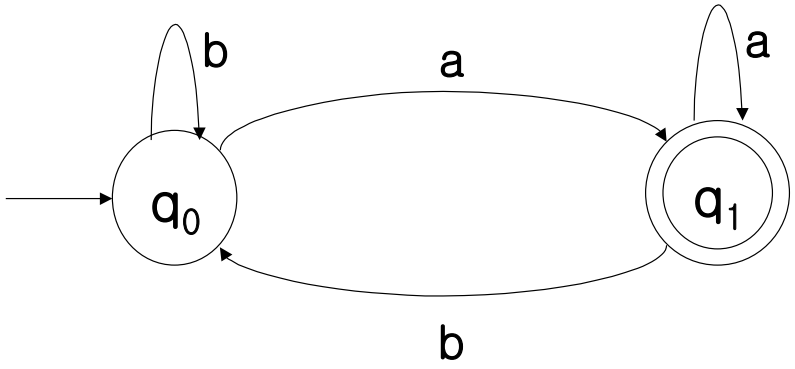
- (q_o, w) : **초기 형상** , w : 입력스트링
- (q, λ) : **최종 형상** (accepting), $q \in F$
- (q', λ) : **최종 형상** (rejecting), $q' \notin F$

결정적 유한 인식기(dfa : deterministic finite accapter)

- **[ex]** DFA M = ($\{q_0, q_1\}$, {a, b}, δ, q_0, $\{q_1\}$)

  where δ : $\delta(q_0, a) = q_1$ $\delta(q_0, b) = q_0$

  $\delta(q_1, a) = q_1$ $\delta(q_1, b) = q_0$



  <u>입력 스트링 aba</u>

  $(q_0, aba) \vdash (q_1, ba) \vdash (q_0, a) \vdash (q_1, \lambda)$

  초기 형상 : (q_0, aba), 최종 형상 : (q_1, λ)

  $(q_0, aba) \vdash^* (q_1, \lambda)$ ∴ **M accepts aba**

  <u>입력 스트링 abb</u>

  $(q_0, abb) \vdash (q_1, bb) \vdash (q_0, b) \vdash (q_0, \lambda)$

  q_0 is not in F ∴ **M rejects abb**

언어와 결정적 유한 인식기

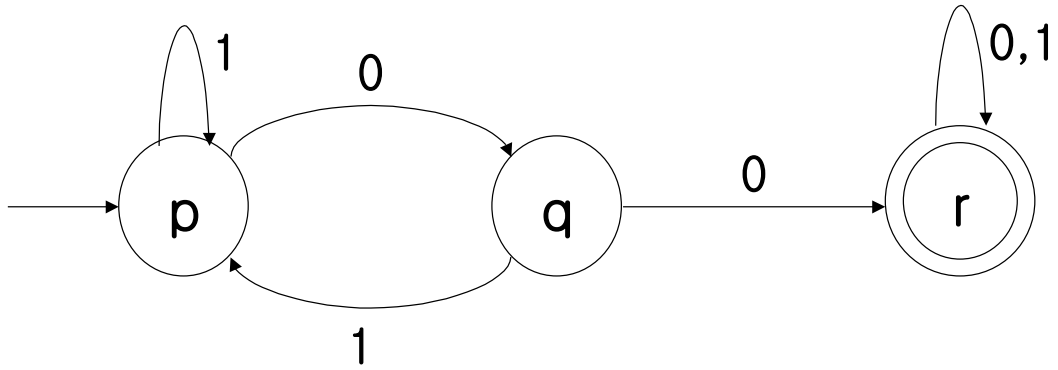
- 언어 : 주어진 오토마타에 의해 승인되는 모든 문자열들의 집합

- 정의 2.2 : dfa $M = (Q, \Sigma, \delta, q_0, F)$에 의해 인식되는 언어란 M에 의해 승인되는 Σ에 대한 모든 문자열들의 집합이다. 공식적인 표현을 사용하면, 다음과 같다:

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}.$$

결정적 유한 인식기(dfa : deterministic finite accapter)

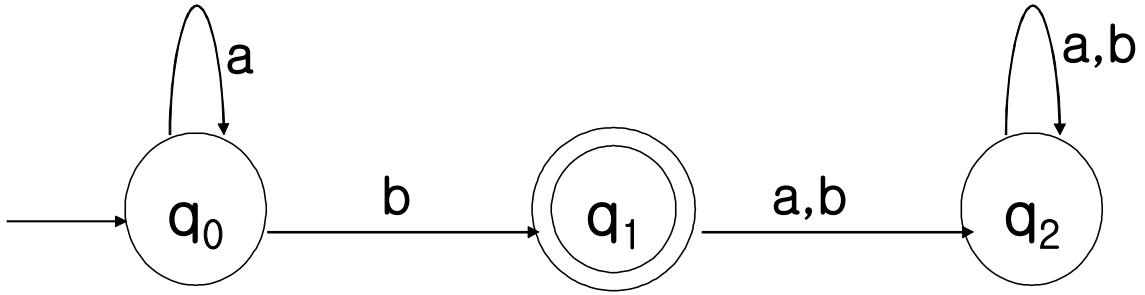
- **[ex]**



- 00을 포함하는 0, 1의 스트링
- 즉, L = { w | w ∈ {0,1}*00{0,1}* }

결정적 유한 인식기(dfa : deterministic finite accapter)

- **예제 2.2(48 page)**

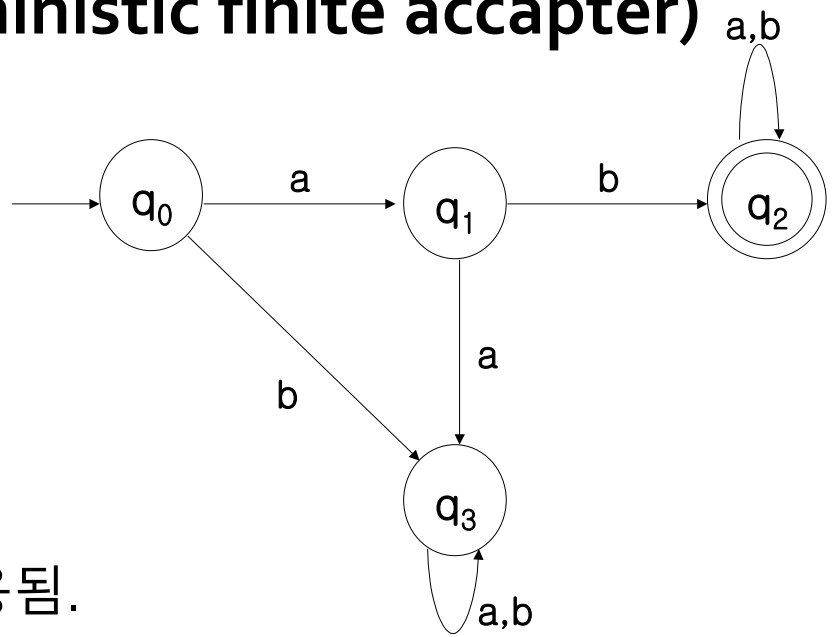


  - $L(M) = \{ a^n b \mid n \geq 0 \}$

- 트랩 상태(trap state) : 전이하고 이후 그 상태를 벗어날 수 없게 되는 상태

결정적 유한 인식기(dfa : deterministic finite accapter)

- **예제 2.3(50 page)**



입력 스트링 aab

$(q_0, aab) \vdash (q_1, ab) \vdash (q_3, b) \vdash (q_3, \lambda)$

$(q_0, aab) \vdash^* (q_3, \lambda)$ ∴ **M rejects aab**

✔ 하나의 edge 상에 여러 개의 symbol이 사용됨.

✔ q_3 : **trap 상태** : 다른 상태로의 전이가 더 이상 불가능한 상태.
(최종 상태는 trap 상태가 아니다.)

✔ trap 상태 및 그와 연결된 edge들을 지워도 됨

입력 스트링 aab

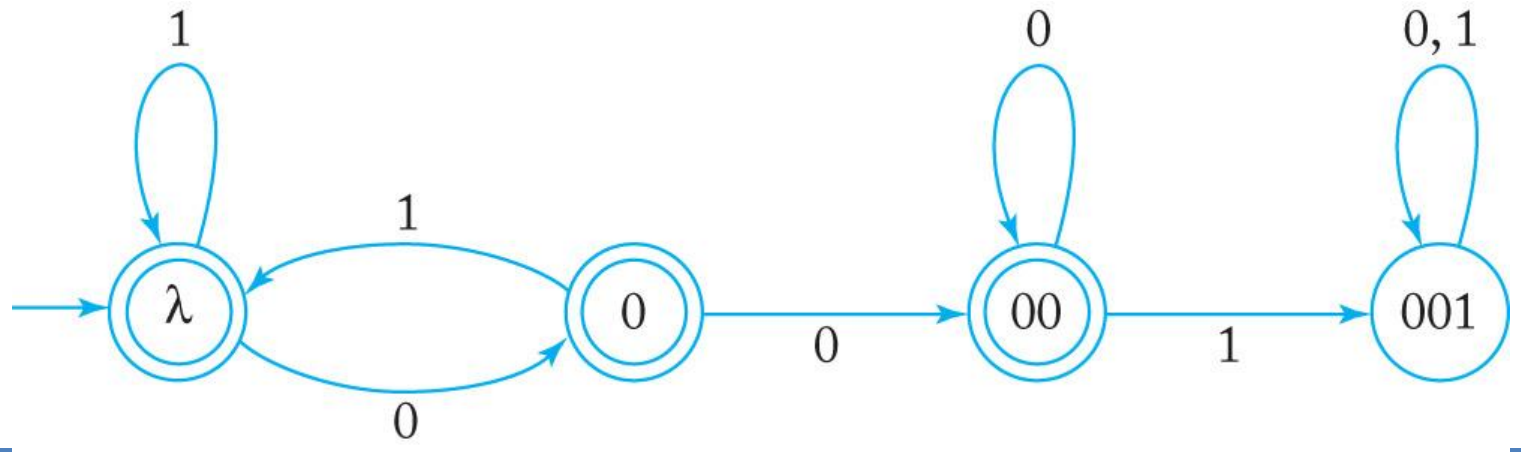
$(q_0, aab) \vdash (q_1, ab) \vdash$

전이할 상태가 존재하지 않는다. ∴ **M rejects aab**

결정적 유한 인식기(dfa : deterministic finite accapter)

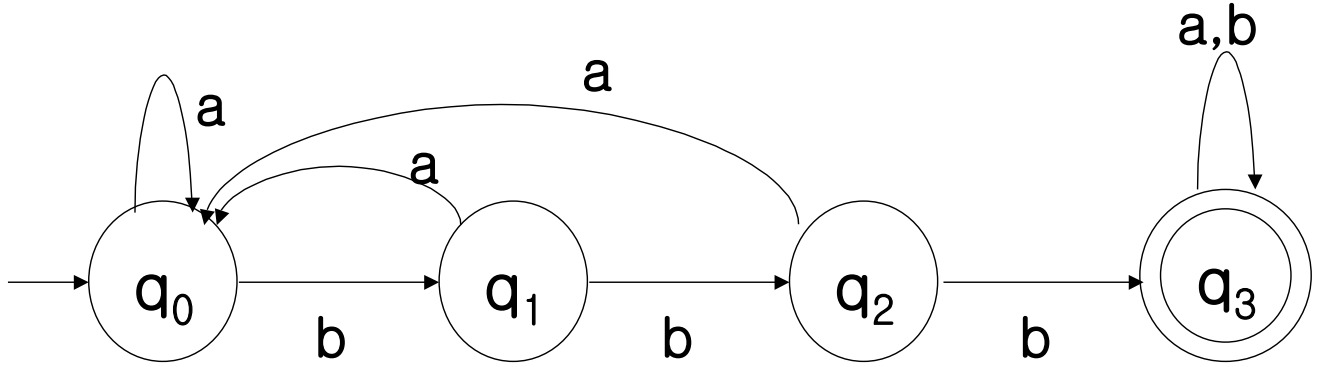
- **예제 2.4(51 page)**
- 부문자열 001이 나타나는 지 확인하기
  - 두 개의 00 선행 : 상태 라벨 00
- 입력문자열 001 : 거부
  - 비승인 상태 : 001 라벨
  - 트랩

$$\delta(00, 0) = 00$$

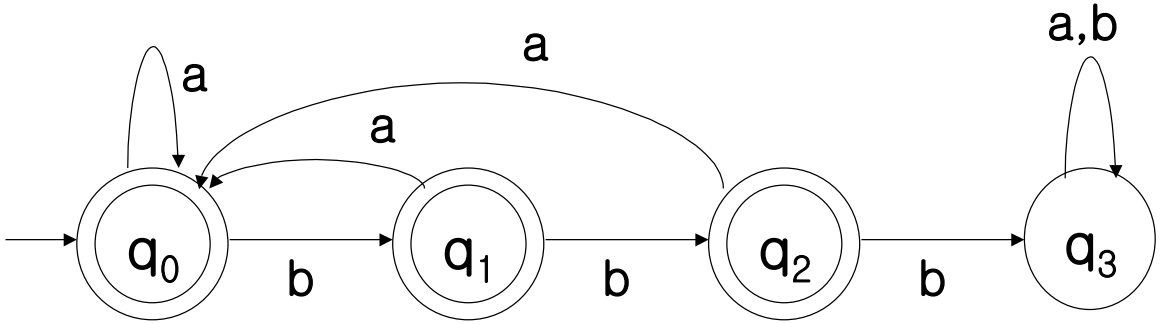


결정적 유한 인식기(dfa : deterministic finite accapter)

- [예]



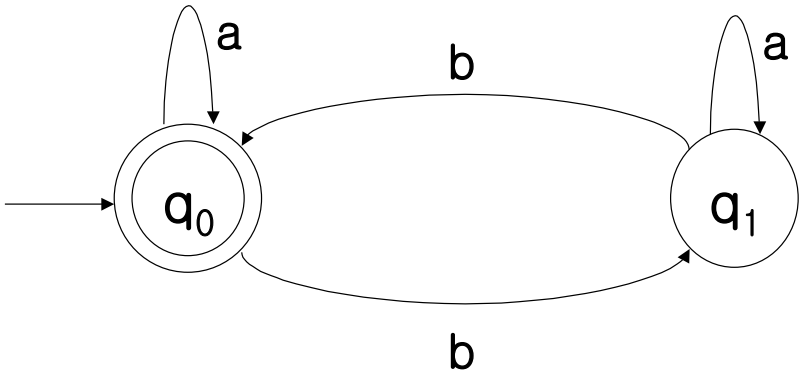
  - 부분스트링 bbb를 포함하는 스트링을 인식하는 dfa M



  - M의 여집합(부분스트링 bbb를 포함하지 않는 스트링을 인식하는 dfa M)

결정적 유한 인식기(dfa : deterministic finite accapter)

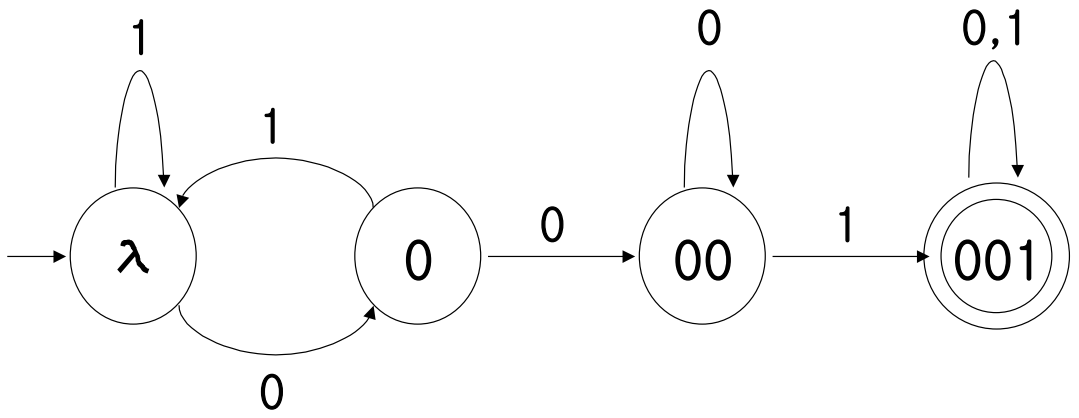
- [예]



  - 짝수 개의 b를 가진 스트링을 인식하는 dfa

결정적 유한 인식기(dfa : deterministic finite accapter)

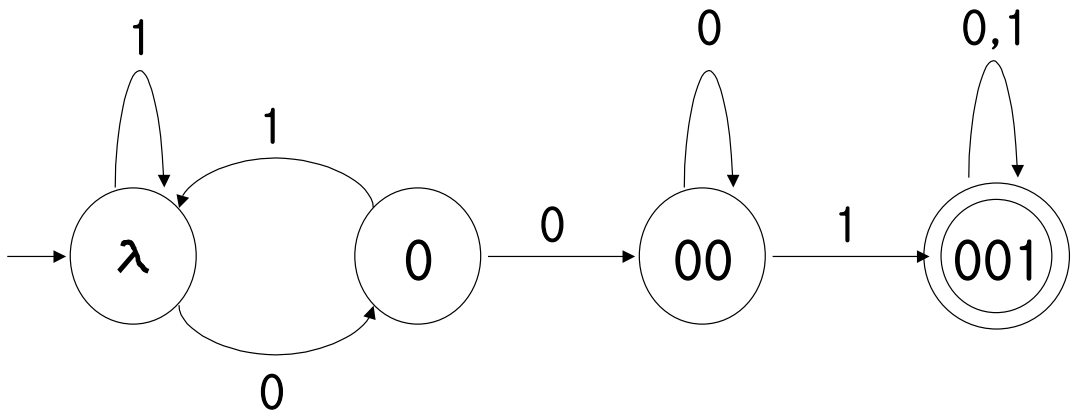
- [예]



  - 부분스트링 001을 포함하는 스트링을 인식하는 dfa

결정적 유한 인식기(dfa : deterministic finite accapter)

- [예]



  - 부분스트링 001을 포함하는 스트링을 인식하는 dfa

정규언어

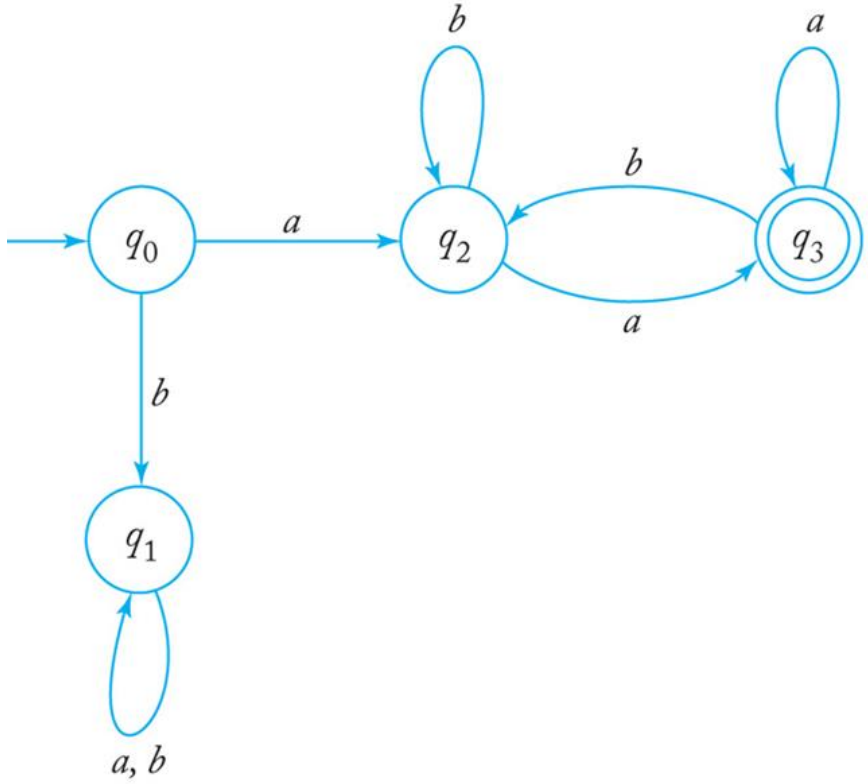
- 모든 유한 오토마타는 특정 언어 인식
- 언어군(family) : 모든 가능한 유한 오토마타들과 관련된 언어들의 집합

- 정의 2.3
  - 언어 L에 대하여, L = L(M)을 만족하는 결정적 유한 인식기 M이 존재하고 오직 그럴 때에만 L을 정규 언어(regular language)라 부른다.

- $L = \{(ab)^n a, n > 0\}$

정규언어

- **예제 2.5 (53page)** L = { awa | w ∈ {a,b}* } 이 정규언어인가?
  - L을 인식하는 결정적 유한 인식기(dfa)를 만들어 보이면 된다.



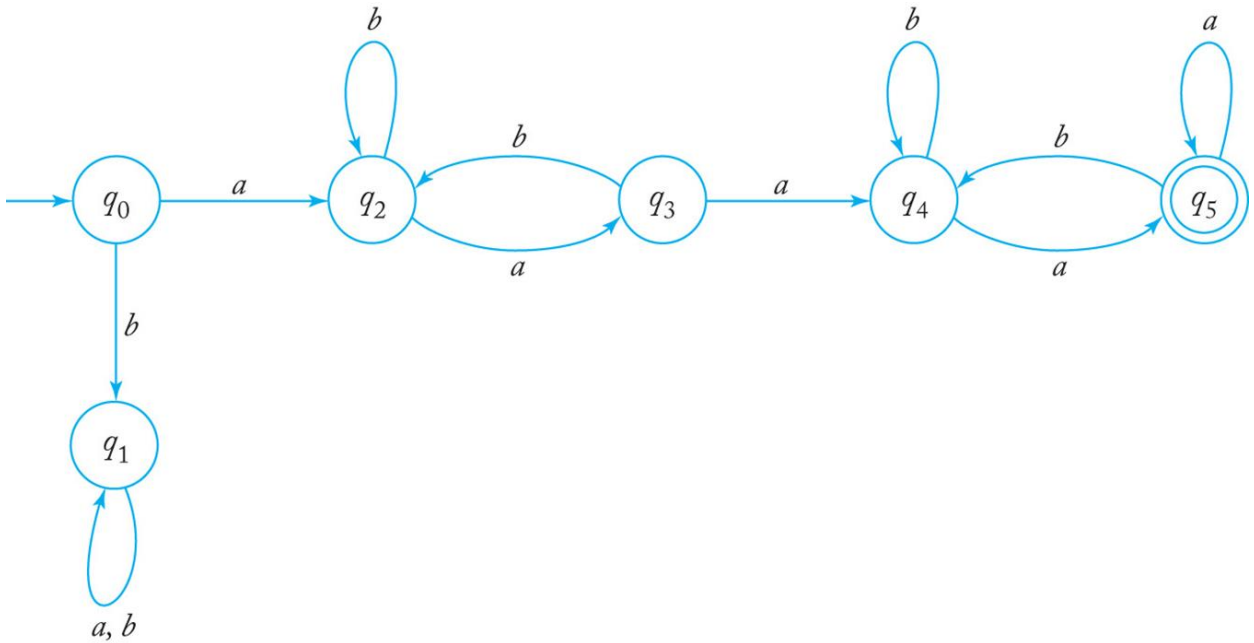
정규언어

- **예제 2.6 (54page)** L = { awa | w ∈ {a,b}* } 이 정규언어일때

$$L^2 = \{aw_1aaw_2a : w_1, w_2 \in \{a,b\}^*\}$$

가 정규 언어임을 보여라.

이 언어를 인식하는 dfa를 구성하기 위해서는 같은 형태를 갖는 두 개의 연속된 문자열을 인식하는 dfa가 필요함



비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 비결정적 유한 상태 자동장치(NFA) 또는 비결정적 유한 오토마타(Non deterministic Finite Automata, NFA)
  - 비결정 : 상태의 입력에 대하여 전이되는 다음 상태가 한 개가 아닌 여러 개이거나 없을 수도 있음
  - 유한 오토마타(인식기)가 비결정적으로 작동하도록 허락하는 경우 표현은 매우 복잡함
  - 비결정성(nondeterminism) : 오토마타의 이동(전이)에 있어 선택을 할 수 있음을 의미
    - 각 상태에서 유일한 이동만을 규정하지 않고 가능한 여러 이동들의 집합을 허용, 오토마타의 전이함수가 상태들의 집합을 치역(range)으로 갖게 하여 가능해짐

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 비결정적 유한 인식기 (nfa)
  - 결정적 유한 인식기 (dfa)와 유사하게 정의하지만
    - 입력기호를 사용하지 않고 바로 다음 상태로 전이할 수 있음을 나타내는 ϵ –전이($\epsilon - transition$) 또는 λ-전이 ($\lambda - transition$)를 추가

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 상태 전이 함수 $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$

  $(\Sigma \cup \{\lambda\})$: 입력 가능한 모든 문자열(Σ)과 길이가 0인 문자열(λ)의 집합

  - 함수의 두 번째 인자로 빈 문자열(λ)을 넣을 수 있음
  - 예) $\delta(q_1, \lambda)$: 입력 장치가 문자를 읽지 않아도 상태 전이가 가능하다는 것을 의미 - DFA에서의 입력장치는 오른쪽으로 만 움직였는데 NFA는 멈춰 있을 수도 있음

  2^Q : 집합 Q 의 모든 부분 집합의 집합, 정의되지 않은 전이 상태(공집합)로 갈 수도 있음

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 입력 기호에 대해서 λ -전이에 의해 0개 이상의 이동 가능
  - 어떤 nfa가 q_1 상태에서 a를 입력 받았을 때 q_0, q_2가 다음 상태로 가능
    - $\delta(q_1, \lambda) = \{q_0, q_2\}$로 표현
- dfa와 마찬가지로 비결정적 유한 인식기도 전이 그래프로 표현
  - 전이 그래프의 정점들은 Q 에 의해 결정되며, $\delta(q_i, x)$ 가 q_j를 포함할 경우만 라벨 x를 가지는 간선 (q_i, q_j) 가 존재
    - x 는 빈 문자열일 수도 있으므로 라벨 λ 를 갖는 간선들도 존재할 수 있음

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- nfa에서의 인식과 거부
  - 인식(accept) : 어떤 문자열에 대해, 이 문자열이 모두 처리된 후에 자동장치(automata)를 종료 상태에 놓이게 하는 이동 순서가 존재
  - 거부(reject) : 가능한 다음 상태의 경우가 없을 때
- **비결정성**은 nfa가 모든 문자열들을 인식하고자 한다는 가정
  - 각 상태에서 최상의 이동 선택 : 직관적인 통찰력 필요

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- [정의] **비결정적 유한 인식기**(nfa : nondeterministic finite accepter)
  - $M = (Q, \sum, \delta, q_0, F)$,
  - 상태전이함수 $\delta : Q \times (\sum \cup \{\lambda\}) \rightarrow 2^Q$, 나머지는 dfa 와 동일

(1) Q : 내부 상태(internal state)들의 유한집합

(2) $\sum$: 입력기호(input symbol)들의 유한 집합, 입력 알파벳(input alphabet)

(3) $\delta : Q \times (\sum \cup \{\lambda\}) \rightarrow 2^Q$: 전체 함수(total function), 전이함수(transition function), 상태 전이 함수(state transition function)

(4) $q_0 \in Q$: 초기 상태(initial state)

(5) $F \subseteq Q$: 승인 상태(final state), 최종 상태의 집합

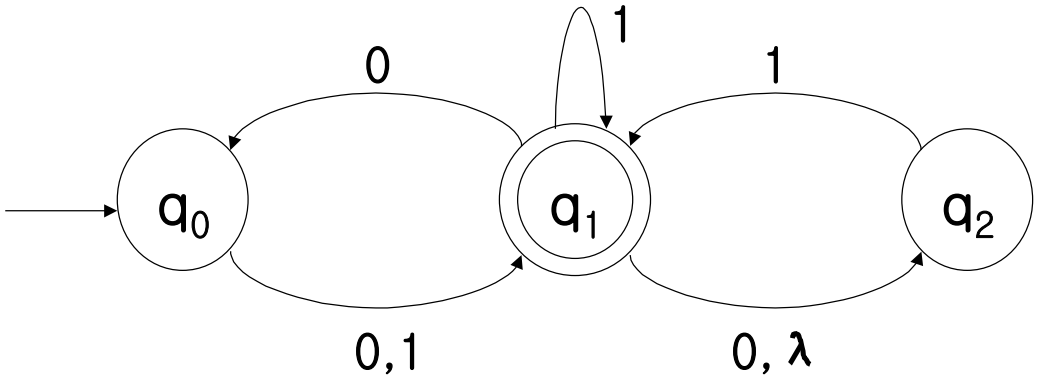
$$\delta(q_1, a) = \{q_0, q_2\}$$

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- **<u>nfa의 특성</u>** :
  - 주어진 상태와 입력기호에 대해 o개 이상의 상태로 move가 가능
    - $\delta(q_i, a) = \Phi$ 가 가능
    - $\delta(q_i, a) = \{q_j, q_k\}$ 가 가능
    - $\delta(q_i, \lambda) = q_j$ 가 가능 : **λ-transition**
  - δ는 전역함수(total function)가 아니다.
  - 입력 스트링을 다 읽은 후 최종상태에 도달하면
    - **'입력 스트링은 NFA에 의해 인식된다'**고 한다.

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- **[ex]** nfa의 특성



$\delta(q_1, o) = \{q_0, q_2\}$

✔ $\delta(q_1, \lambda) = \{q_2\}$: **λ-transition**

✔ $\delta(q_2, o) = \Phi$

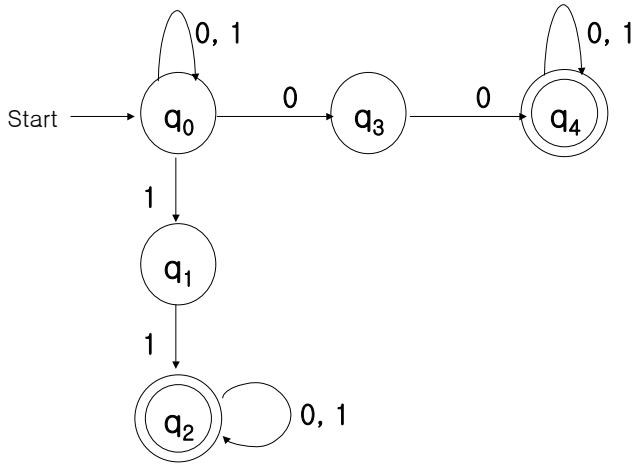
입력 스트링 101 **NFA accepts 101**

입력 스트링 110 **NFA rejects 110**

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- **[ex]**

$(q_0, 01001)$
$\vdash (q_0, 1001)$
$\vdash (q_0, 001)$
$\vdash (q_3, 01)$
$\vdash (q_4, 1)$
$\vdash (q_4, \lambda)$ $\therefore 01001 \in L(M)$



nfa δ		0	1
초기	q_0	$\{q_0,q_3\}$	$\{q_0,q_1\}$
	q_1	Φ	$\{q_2\}$
최종	q_2	$\{q_2\}$	$\{q_2\}$
	q_3	$\{q_4\}$	Φ
최종	q_4	$\{q_4\}$	$\{q_4\}$

nfa δ		0	1
초기	q_0	$\{q_0,q_3\}$	$\{q_0,q_1\}$
	q_1	-	q_2
최종	q_2	q_2	q_2
	q_3	q_4	-
최종	q_4	q_4	q_4

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- **[ex]** nfa M = ($\{q_0, q_1, q_2, q_3, q_4\}$, $\{1,2,3\}$, δ, q_0, $\{q_4\}$)

nfa δ	1	2	3
초기 q_0	$\{q_0,q_1\}$	$\{q_0,q_2\}$	$\{q_0,q_3\}$
q_1	$\{q_1,q_4\}$	q_1	q_1
q_2	q_2	$\{q_2,q_4\}$	q_2
q_3	q_3	q_3	$\{q_3,q_4\}$
최종 q_4		- - -	

$(q_0, 12321) \vdash (q_1, 2321) \vdash (q_1, 321) \vdash (q_1, 21) \vdash (q_1, 1) \vdash (q_4, \lambda)$

$\therefore 12321 \in L(M)$

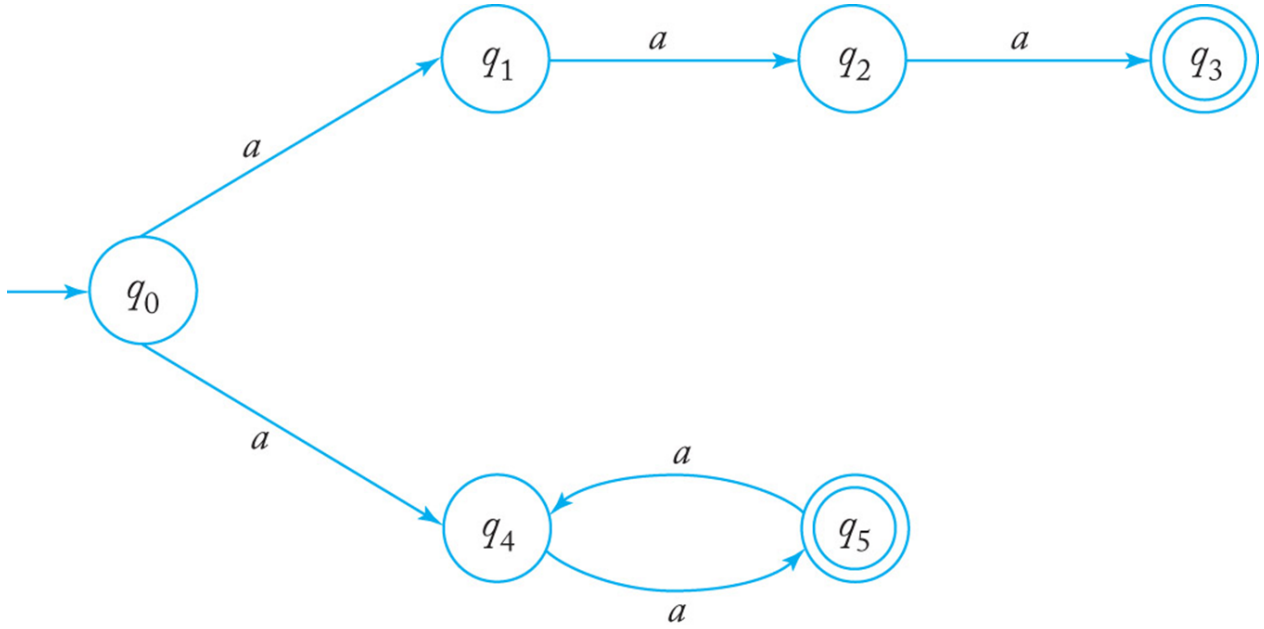
비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 예제 2.7(59 page)

전이 그래프 : q_0로 부터의 라벨 a를 갖는 전이가 두 개

- 비결정적 유한 인식기

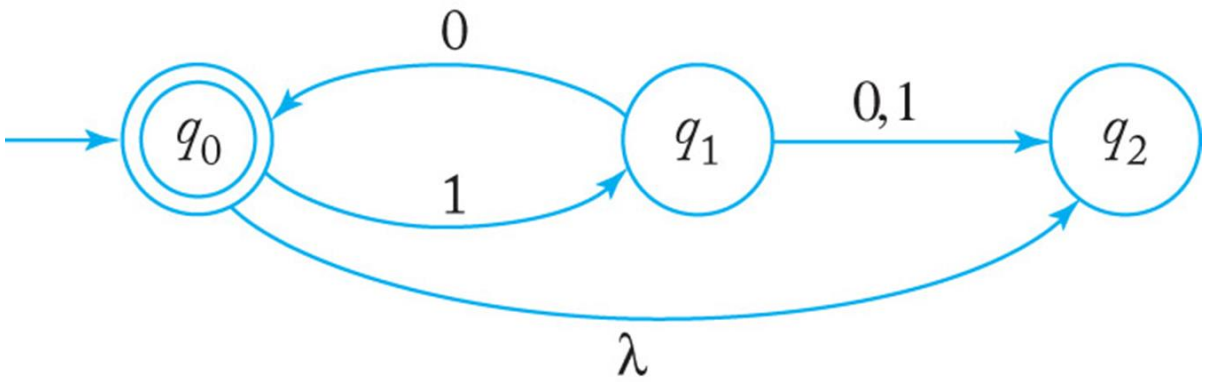
$\delta(q_0, a) = \{ q_1, q_4 \}$



비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 예제 2.8(59 page) 이 오토마타는 하나의 정점으로부터 같은 라벨을 갖는 여러 개의 간선들이 있을 뿐 아니라, 또한 λ-전이를 가지므로 비결정적

$\delta(q_0, \lambda) = \{ q_2 \}$ and $\delta(q_2, o) = \varnothing$



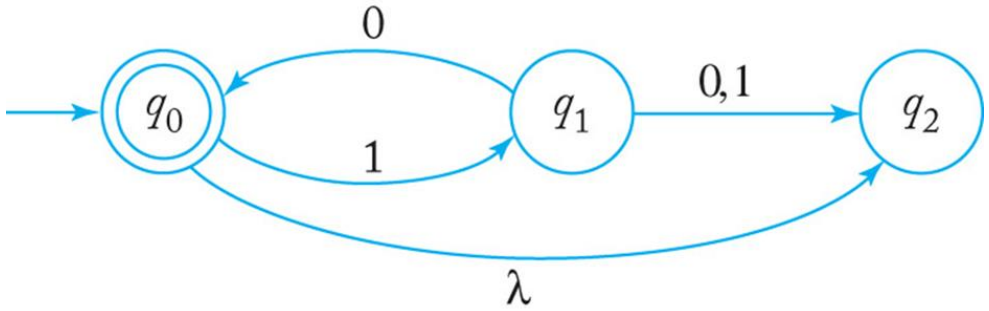
비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- dfa에서와 마찬가지로 nfa에서도 전이 함수는 두 번째 인수로 문자열을 갖도록 확장될 수 있음
- 확장 전이 함수 δ*

$$\delta^*(q_i, w) = Q_j$$

- 예제 2.8에서 확장 전이 함수

$\delta^*(q_0, 10) = \{ q_0, q_2 \}$ $\delta^*(q_0, 101) = \{ q_1 \}$

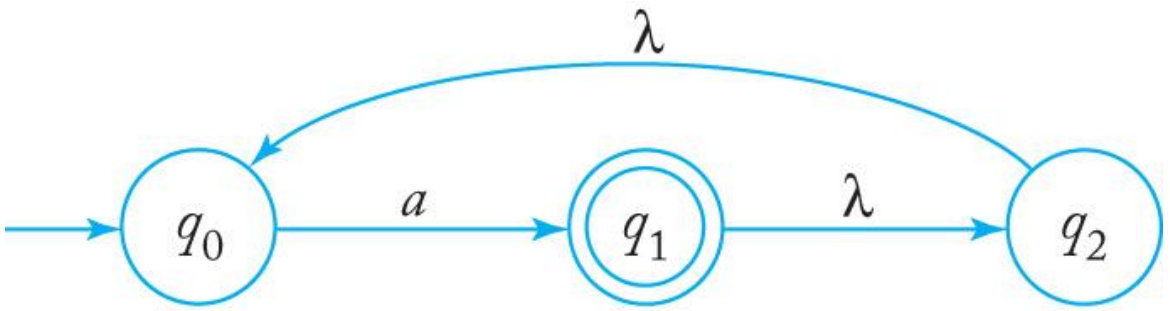


비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 정의 2.5
  - nfa의 확장 전이 함수는 다음과 같이 정의한다.
    - 전이 그래프에서 q_i로부터 q_j로의 라벨 w를 갖는 보행이 존재하고 오직 그럴 때에만 $\delta^*(q_i, w)$가 q_j를 포함한다. 이는 모든$q_i, q_j \in Q$와 $w \in \Sigma^*$에 대해 성립한다.

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 예제 2.9(60 page)
  - 여러 개의 λ-전이
  - $\delta(q_2, a)$: 정의 없는 전이
  - $\delta^*(q_1, a), \delta^*(q_2, \lambda)$ 구하기



$$\delta^*(q_1, a) = \{q_0, q_1, q_2\}$$

$$\delta^*(q_2, \lambda) = \{q_0, q_2\}$$

$$\delta^*(q_2, aa) = \{q_0, q_1, q_2\}$$

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

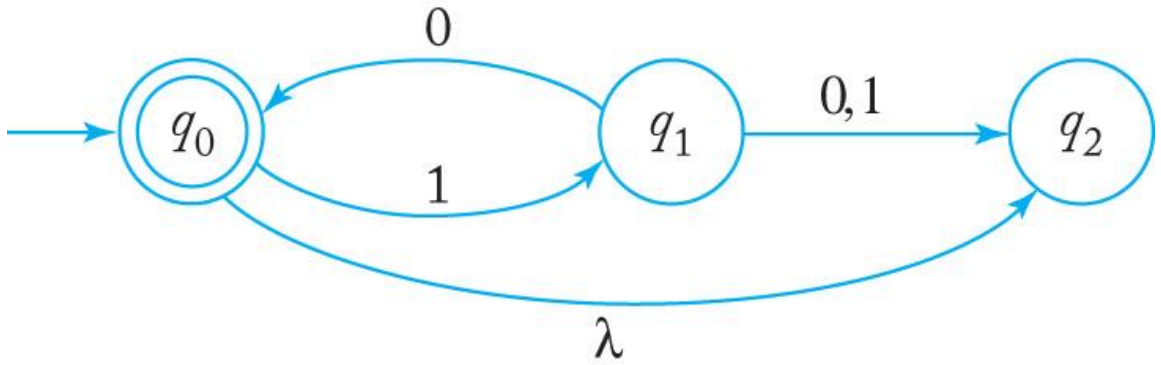
- 정의 2.6
  - nfa에 의해 인식되는 언어

$$L(M) = \{w \in \Sigma^*: \delta^*(q_0, w) \cap F \neq \emptyset\}$$

    - 전이 그래프의 초기 정점에서 승인 정점까지 라벨이 w인 보행이 존재하는 모든 문자열 w로 구성

비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- 예제 2.10(62 page) 그림의 오토마타에서 인식되는 언어는?



  - 승인상태 도달 : 10 반복, 빈문자열
  - $L = \{(10)^n : n \geq 0\}$

    $w = 110, \delta^*(q_0, 11) = q_2$
  - 종말현상(dead configuration)

$$\delta^*(q_0, 110) = \emptyset$$

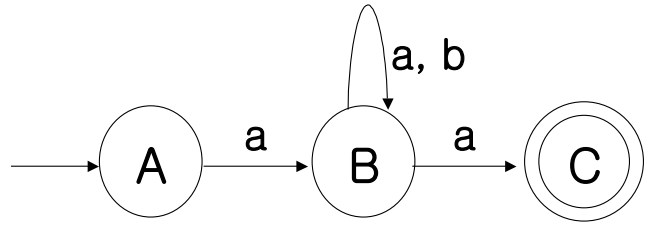
비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- **비결정성의 필요성**
  - nfa를 이용하는 것이 dfa를 이용하는 것보다 훨씬 쉬움
  - dfa로 디자인하기 어려울 경우 먼저 nfa를 디자인 하고 그것과 동치인 dfa로 변환
  - 비결정성은 복잡한 문제를 간략하게 기술하는데 매우 효과적

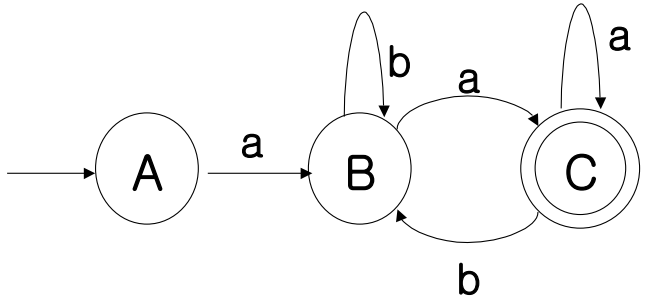
비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- **[ex]**
  - a{a,b}*a 의 nfa와 dfa

NFA :



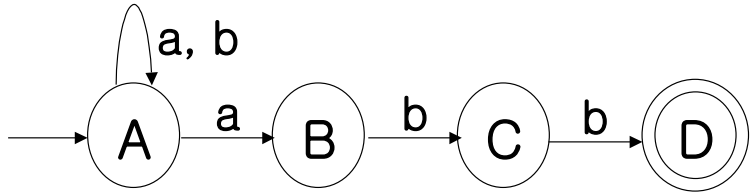
DFA :



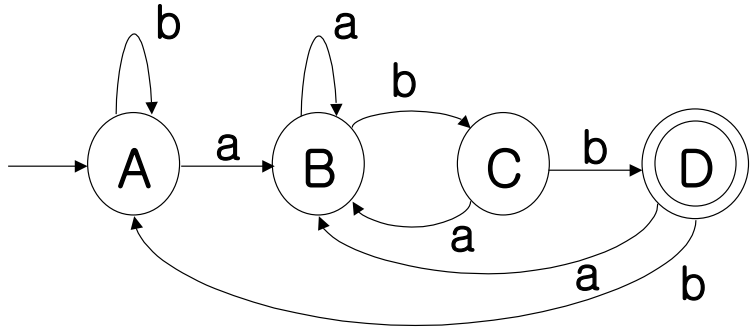
비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- {a,b}*abb 의 nfa와 dfa

NFA :



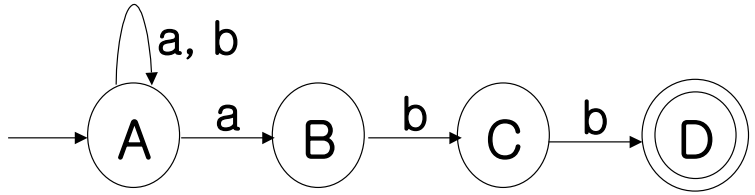
DFA :



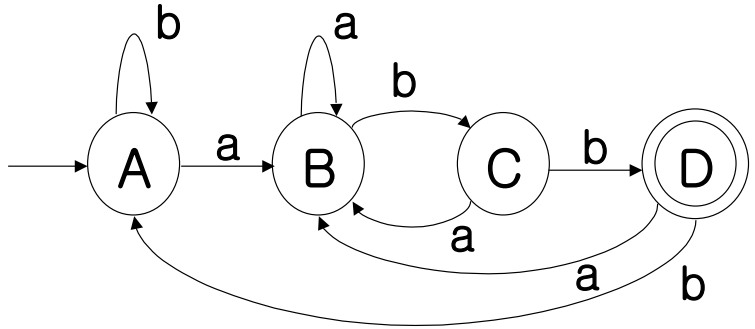
비결정적 유한 인식기
(nfa : nondeterministic finite accepter)

- {a,b}*abb 의 nfa와 dfa

NFA :



DFA :

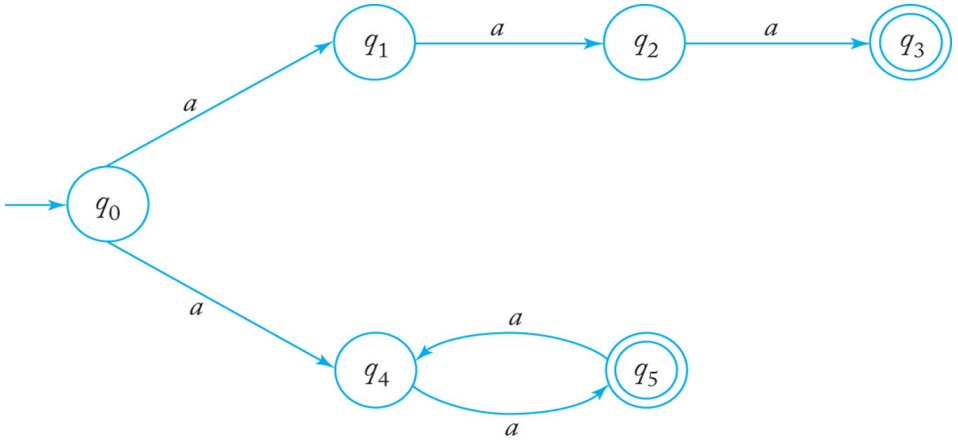


비결정성을 사용하는 이유

- 결정적 알고리즘 : 어떤 단계에서 하나의 선택, 결정
  예) 게임 프로그램
  - 게임 프로그램의 각 단계에서 최적의 이동을 알 수 없는 경우
    - 백트래킹(backtracking) 등의 기법으로 모든 경우를 탐색
    - 가능한 선택 대상이 여러 개 있는 경우 : 하나를 선택하고 그 선택이 최적인지 아닌지가 명확히 밝혀질 때까지 계속 검사
      - 선택이 최적이 아닐 경우 : 마지막 결정 지점으로 되돌아와서 다른 선택에 대한 검사 다시 시작
    - 최적의 선택을 할 수 있는 비결정적 알고리즘을 사용할 경우: 백트래킹 없이 문제를 해결할 수 있으며, 결정적 알고리즘은 추가적인 작업을 통하여 비결정성을 시뮬레이트(simulate)할 수 있음
  - 비결정적 기계 : 탐색-백트랙 알고리즘(search-and-backtrack algorithm)에 대한 모델로 사용될 수 있음

비결정성을 사용하는 이유

- 비결정성은 문제들을 쉽게 해결하는 데 유용함



- 이 nfa가 선택을 내려야 하는 곳은 분명
  - 하나의 선택을 할 경우 문자열 a^3 승인
  - 다른 선택을 할 경우 : 짝수 개의 a를 갖는 모든 문자열들이 승인됨
- 이 nfa에 의해 인식되는 언어 : $\{a^3\} \cup \{a^{2n} : n \geq 1\}$
  - 이 언어는 서로 다른 두 개 집합들의 합집합
  - 이 경우 비결정성은 시작에 필요한 결정을 할 수 있게 함
  - 이 문제에 대한 결정적 해결법 : 정의에서처럼 명확하지 않음

비결정성을 사용하는 이유

- 비결정성은 몇몇 복잡한 언어들을 간명하게 정의하는 데에 효과적
  - 문법의 정의에서 비결정적 요소를 가지고 있음을 유의
    - 생성규칙 $S \rightarrow aSb \mid \lambda$
      - 모든 시점에 두 생성규칙들 중 하나를 선택
      - 단지 두 개의 규칙으로 많은 문자열들을 지정할 수 있게 해줌

- 비결정성을 도입하는 데에는 기술적인 이유 존재
  - dfa보다는 nfa에 대한 어떤 이론적인 결과를 더 쉽게 수립하게 됨
    - 이 두 가지 형태의 오토마타 사이에는 근본적인 차이가 존재하지 않음
    - 비결정성을 사용하는 것이 결론의 일반성을 그대로 유지하면서 공식적인 논증을 간단히 하는 효과를 보이게 됨

결정적 유한 인식기와 비결정적 유한 인식기의 동치성

- 오토마타들간의 동치성(equivalence)

[정의] 다음과 같은 조건이 만족되는 경우:

$$L(M_1) = L(M_2)$$

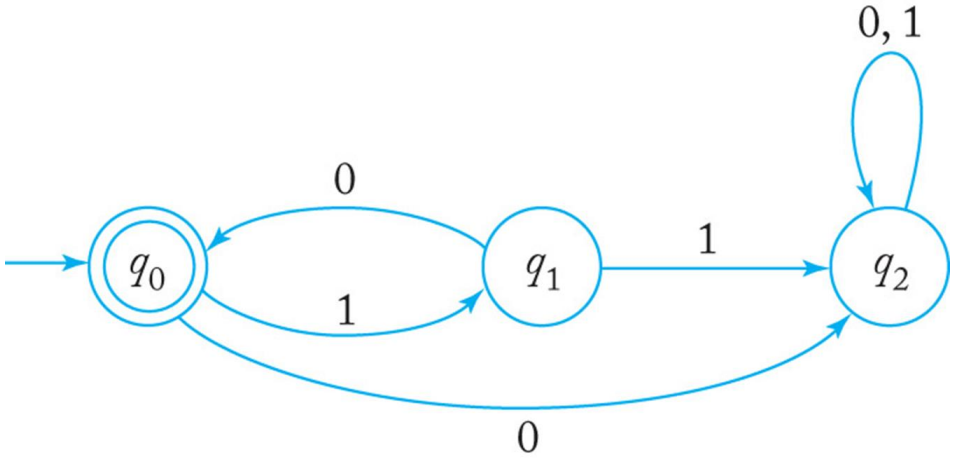
즉, 두 오토마타가 같은 언어를 인식하는 경우, 두 개의 유한 인식기 M_1과 M_2는 동치

결정적 유한 인식기와 비결정적 유한 인식기의 동치성

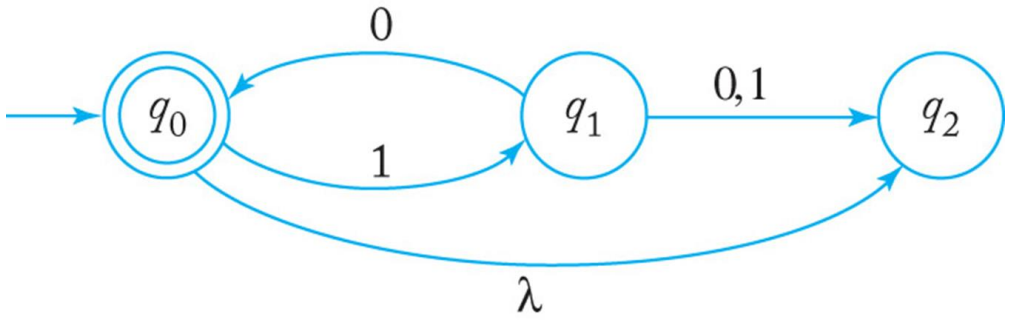
- **예제 2.11(66 page)** 다음의 dfa와 nfa는 서로 동치

두 인식기 모두 언어 $\{(10)^n : n \geq 0\}$을 인식

dfa :



nfa :



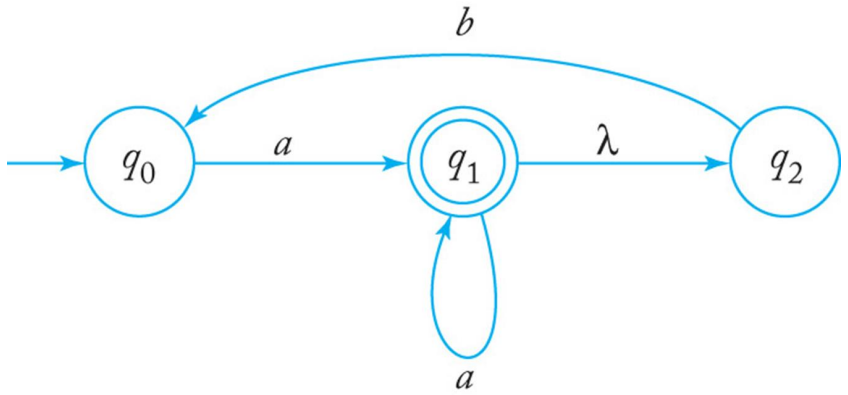
결정적 유한 인식기와 비결정적 유한 인식기의 동치성

- 강력(powerful) : 한 종류의 오토마타가 다른 종류의 오토마타에서 수행할 수 없는 어떤 일을 수행할 수 있을 때
- dfa : nfa의 한 제한된 종류
  - dfa에 의해 인식되는 모든 언어 : 어떤 nfa에 의해서도 인식될 수 있음
  - nfa에 의해 인식되는 모든 언어들에 대해 같은 언어를 인식하는 dfa가 항상 존재
- 문자열 *w*를 읽은 후
  - 각 상태에 라벨 부여
    - 동등한 dfa가 라벨이 $\{qi, qj, \ldots, qk\}$인 상태에 놓이도록
    - dfa의 각 상태의 라벨이 nfa의 상태들의 집합이 되도록
- $|Q|$개 상태들의 집합에 대하여, $2^{|Q|}$개의 부분집합들이 존재
  - 대응되는 dfa에는 유한개의 상태들이 존재

임의의 nfa를 동치인 dfa로 구성

- **예제 2.12(67 page)** 다음 그림의 nfa를 동치인 dfa로 변환
  - 상태 q_0에서 시작 : dfa의 초기 상태에 라벨 $\{q_0\}$를 부여
  - 문자 a를 읽은 후, 이 nfa는 상태 q_1에 놓이거나 또는 λ-전이를 한 후 상태 q_2에 놓임
    - 해당 dfa는 라벨이 $\{q_1, q_2\}$인 상태와 다음과 같은 전이를 가져야 함 : $\delta(\{q_0\}, a) = \{q_1, q_2\}$
  - 상태 q_0에서 심벌 b가 입력되었을 때
    - nfa의 상태 전이는 정의되어 있지 않으며, 따라서 해당 dfa는 다음의 전이를 가져야 함 : $\delta(\{q_0\}, b) = \emptyset$

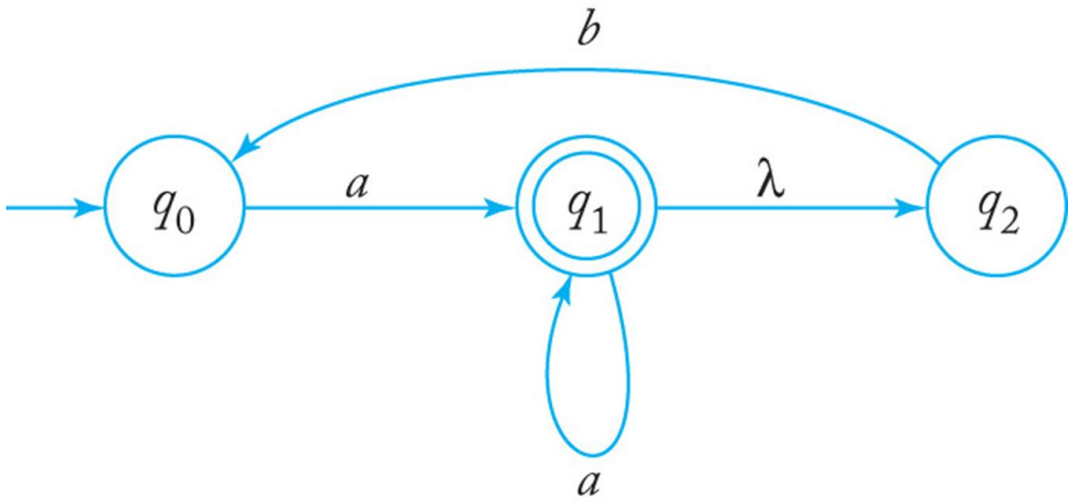
nfa :



임의의 nfa를 동치인 dfa로 구성

- **예제 2.12(67 page)** 계속
  - 라벨 ∅을 갖는 상태는 nfa에서의 불가능한 이동을 표현하는 것
    - 문자열을 승인할 수 없음을 의미
    - dfa에서의 이 상태는 비승인 트랩 상태(nonfinal trapstate)

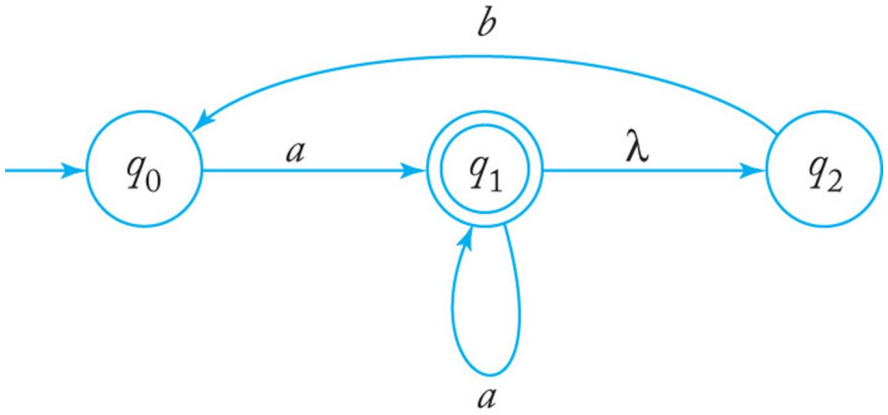
nfa :



임의의 nfa를 동치인 dfa로 구성

- **예제 2.12(67 page)** 계속
  - dfa에 상태 $\{q_1, q_2\}$ 추가 : 이 상태로부터 다른 상태로의 전이 고려
    - dfa에서의 이 상태는 nfa에서 도달할 수 있는 두 개의 상태가 있으므로 nfa를 다시 참조하여야 함
    - nfa가 상태 q_1에 있고 a를 읽을 경우 : 그대로 상태 q_1에 놓임
    - nfa는 상태 q_1에서 λ-전이를 통해 상태 q_2로 전이할 수 있음
      - 같은 입력에 대해, 이 nfa는 상태 q_2에서 어떤 전이도 정의되어 있지 않음 : $\delta(\{q_1, q_2\}, a) = \{q_1, q_2\}$.

nfa :



임의의 nfa를 동치인 dfa로 구성

- **예제 2.12(67 page)** 계속

  $\delta(\{q_1, q_2\}, b) = \{q_0\}$.

  - 각 상태에 대해 모든 전이 정의
    - dfa로 구성

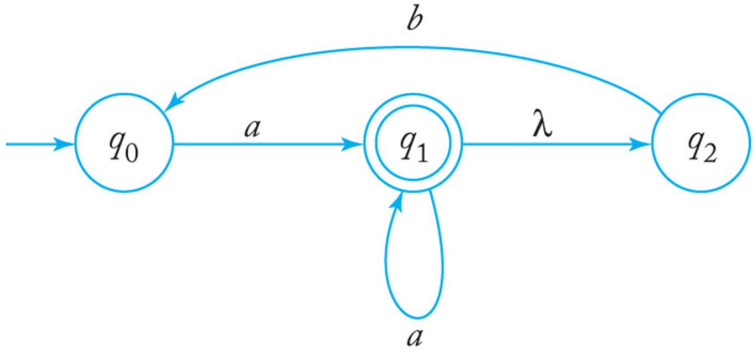
$\delta(\{q_0\}, a) = \{q_1, q_2\}$

$\delta(\{q_0\}, b) = \emptyset$

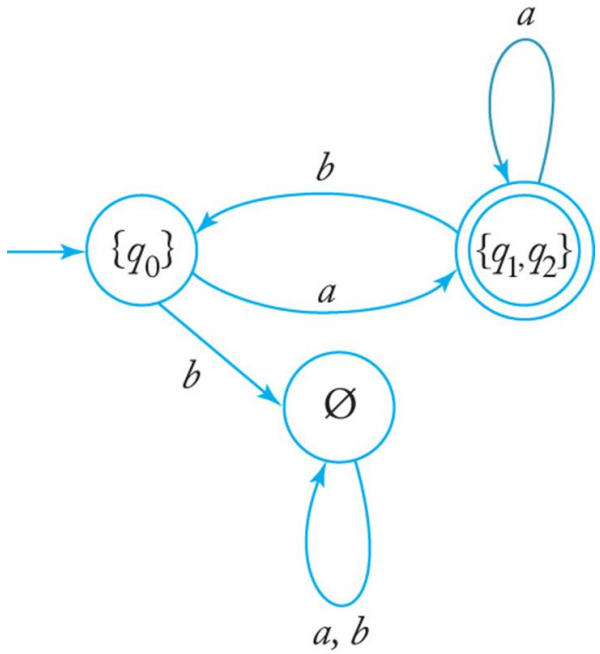
$\delta(\{q_1, q_2\}, a) = \{q_1, q_2\}$

$\delta(\{q_1, q_2\}, b) = \{q_0\}$

nfa :



dfa :



결정적 유한 인식기와 비결정적 유한 인식기의 동치성

[정리 2.2]

언어 L : 비결정적 유한 인식기 $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$에 의해 인식되는 언어

다음을 만족하는 결정적 유한 인식기 $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$가 항상 존재

$$L = L(M_D)$$

procedure: nfa-to-dfa

1. 정점 $\{q_0\}$를 갖는 그래프 G_D를 생성 : 이 정점을 초기 정점으로 함

2. 다음 과정을 모든 간선들이 생성될 때까지 반복:

    Σ의 어떤 원소 a ∈ Σ에 대해 진출 간선을 갖지 않는 G_D의 정점 $\{q_i, q_j, \ldots, q_k\}$를 선택한다. $\delta^*_N(q_i, a)$, $\delta^*_N(qj, a)$, …, $\delta^*_N(qk, a)$들을 계산

    $\delta^*N(qi, a) \cup \delta^*_N(q_j, a) \cup \ldots \cup \delta^*_N(q_k, a) = \{q_l, q_m, \ldots, q_n\}$이라 하고 G_D에 라벨이 $\{q_l, q_m, \ldots, q_n\}$인 정점이 아직 없다면, 이를 라벨로 갖는 정점 추가

    G_D에 정점 $\{q_i, q_j, \ldots, q_k\}$로부터 정점 $\{q_l, q_m, \ldots, q_n\}$으로 향하는 간선을 추가하고 이에 라벨 a를 부여

3. G_D의 상태들 중 라벨이 $q_f \in F_N$을 포함하는 모든 상태들을 승인 상태로 지정

4. 인식기 M_N이 λ를 인식하는 경우, G_D의 정점 $\{q_o\}$를 승인 정점으로 지정

단계 2의 반복작업이 한 번 수행될 때마다 G_D에 간선이 하나씩 추가됨

G_D의 간선의 개수가 많아야 $2^{|Q_N|}|\Sigma|$개이므로 이 반복작업은 유한 시간 내에 종료됨

NFA에서 DFA로의 변환

- [정리2]
  - NFA $M = (Q, \sum, \delta, q_0, F)$에 의해 인식되는 언어를 L이라고 하면
    - L을 인식하는 DFA $M' = (Q', \sum, \delta', q_0', F')$ 은 다음과 같이 구성

(1) Q' : subset of 2^Q , Q'의 한 상태는 $[q_i, q_j, \cdots]$의 형태로 표시

(2) $q_0' = [q_0]$

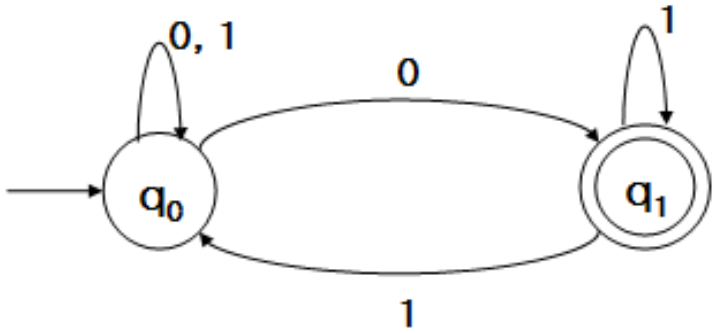
(3) $F' = \{ q \in Q' \mid q \cap F \neq \Phi \}$

(4) $\delta(\{q_1,q_2,\cdots,q_i\}, a) = \{p_1,p_2,\cdots,p_i\} \Leftrightarrow \delta'([q_1,q_2,\cdots,q_i],a) = [p_1,p_2,\cdots,p_i]$

NFA에서 DFA로의 변환

- **[ex]**
  - 정리2를 이용하여 주어진 nfa M을 동치인 dfa M'로 바꾸어 보자.

nfa M	0	1
초기 q_0	$\{q_0, q_1\}$	$\{ q_0 \}$
최종 q_1	-	$\{q_0, q_1\}$



(1) $Q' = \{ [q_0], [q_0, q_1] \}$

(2) 시작상태 $q_0' = [q_0]$

(3) 최종상태 $F' = \{ [q_0, q_1] \}$: NFA의 최종상태 q_1을 포함한 것

(4) $\delta'([q_0],0) = [q_0, q_1]$

$\delta'([q_0],1) = [q_0]$

$\delta'([q_0, q_1],0) = [q_0, q_1]$

$\delta'([q_0, q_1],1) = [q_0, q_1]$

NFA에서 DFA로의 변환

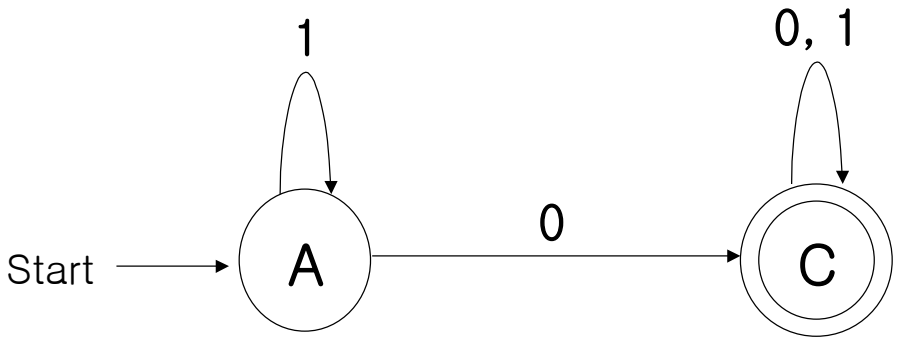
- 상태 전이표

dfa M'	0	1
초기 $[q_0]$	$[q_0, q_1]$	$[q_0]$
최종 $[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

- $[q_0] = A$, $[q_0, q_1] = C$ 로 상태이름을 바꾸고

dfa M'	0	1
초기 A	C	A
최종 C	C	C

- 상태 전이도를 그리면



- 최종의 dfa가 만들어짐