COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

LECTURE 14a :
MIDTERM REVIEW

Definition of DFA

- States
- Alphabet
- Transition function
- Start state
- Accept states

Definition of DFA

- States

- Alphabet
- Transition function
- Start state
- Accept states

Definition of DFA

- States

- Alphabet

a,b,c,d

- Transition function
- Start state
- Accept states

Definition of DFA

- States

- Alphabet

a,b,c,d

- Transition function

- Start state
- Accept states

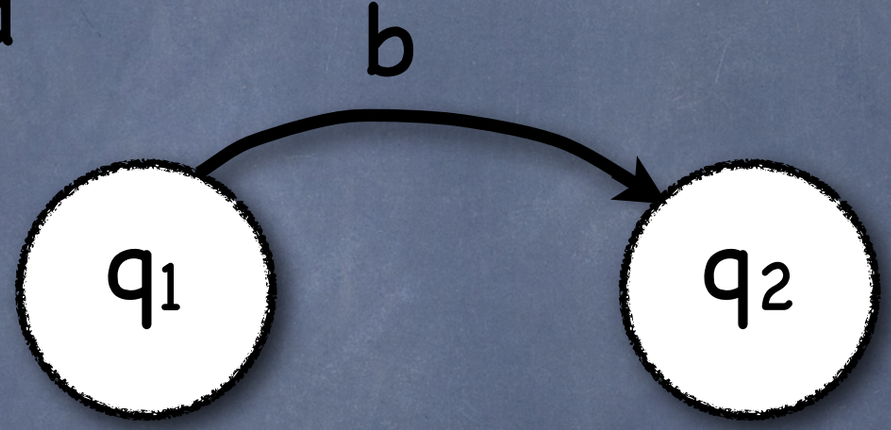
Definition of DFA

- States

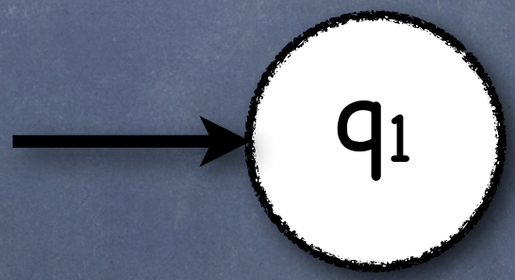
- Alphabet

a,b,c,d

- Transition function



- Start state



- Accept states

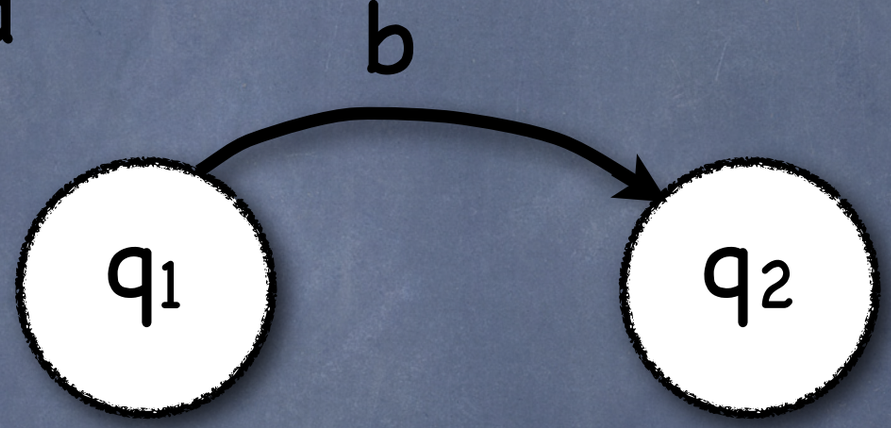
Definition of DFA

- States

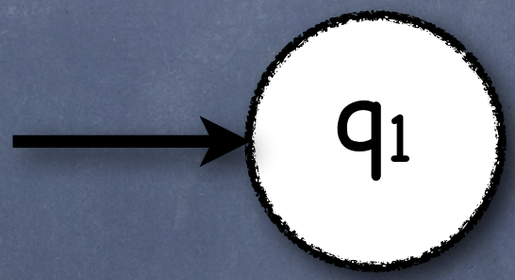
- Alphabet

a,b,c,d

- Transition function



- Start state



- Accept states

Definition of DFA

DEFINITION 1.5

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

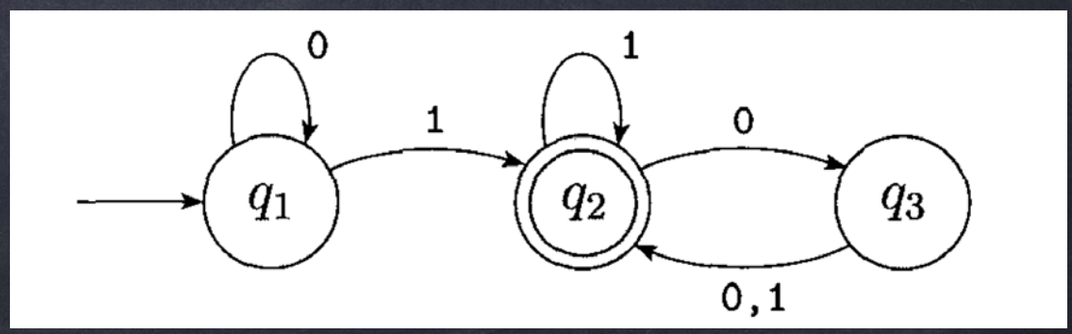
1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Regular Languages

- Let $M=(Q,\Sigma,\delta,q_0,F)$ be a finite state automaton and let $w=w_1w_2...w_n$ $(n\geq 0)$ be a string where each symbol w_i is from the alphabet Σ.

- M **accepts** w if states $s_0,s_1,...,s_n$ exist s.t.
  1. $s_0 = q_0$
  2. $s_{i+1} = \delta(s_i,w_{i+1})$ for $i = 0 \ldots n-1$, and
  3. $s_n \in F$

Regular Languages



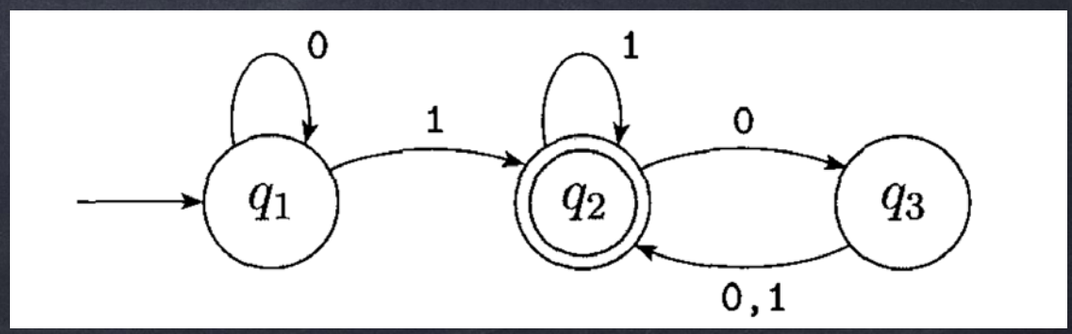
Regular Languages

Regular Languages

Regular Languages

Regular Languages

- M1 **accepts** 10010101 since states $s_0, s_1, \ldots, s_8$ exist s.t.
  - 1. $s_0 = q_1$
  - 2. $s_1 = q_2 = \delta(q_1,1)$, $s_2 = q_3 = \delta(q_2,0)$, $s_3 = q_2 = \delta(q_3,0)$, $s_4 = q_2 = \delta(q_2,1)$, $s_5 = q_3 = \delta(q_2,0)$, $s_6 = q_2 = \delta(q_3,1)$, $s_7 = q_3 = \delta(q_2,0)$, $s_8 = q_2 = \delta(q_3,1)$
  - 3. $s_8 \in F$



Regular Languages

- Let M be a finite state automaton and let $w=w_1w_2...w_n$ ($n \geq 0$) be a string where each symbol w_i is from the alphabet Σ.

- M **recognizes language** A if

$$A = \{ w \mid M \text{ accepts } w \}$$

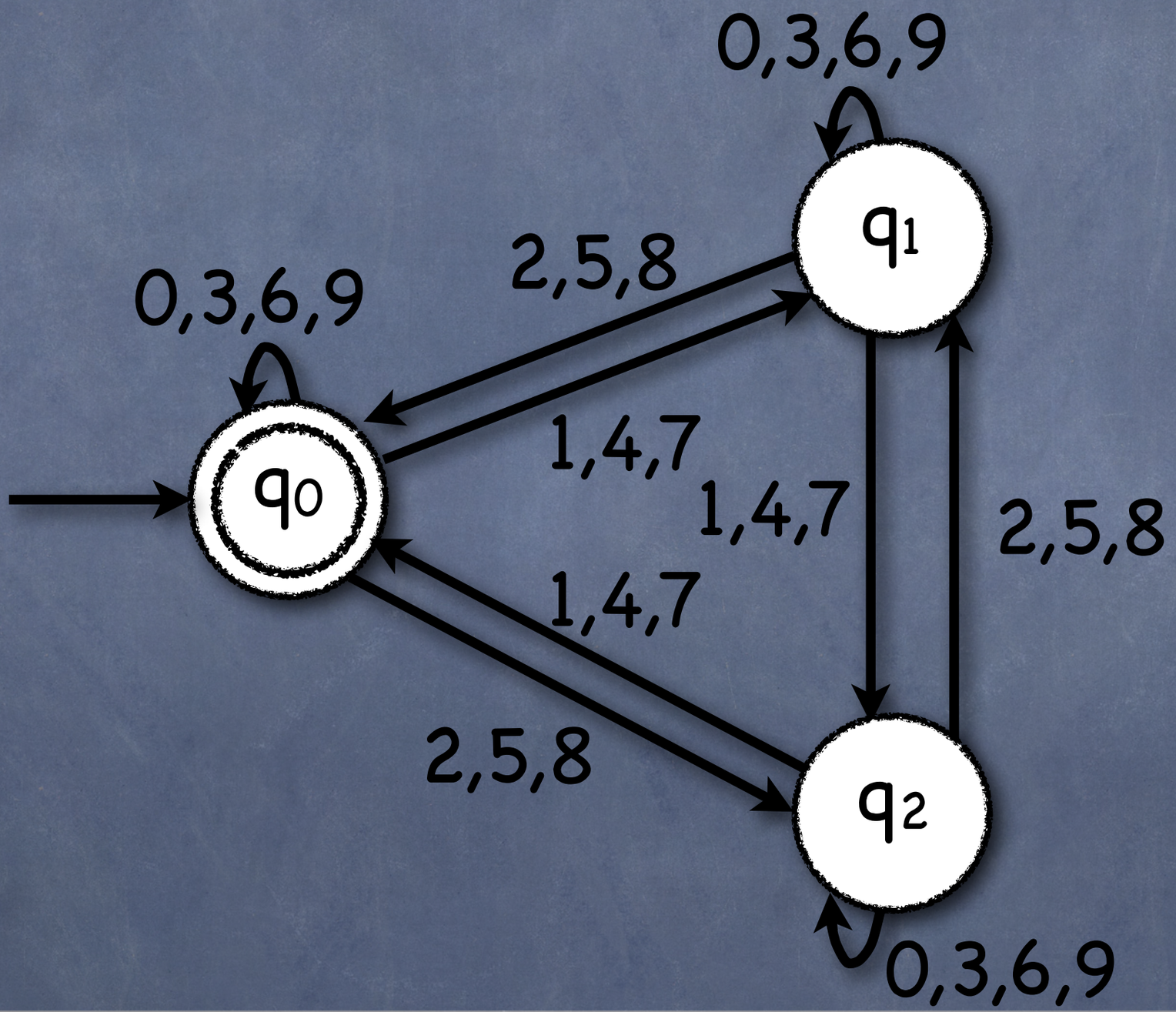
DEFINITION 1.16

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

gcd(B,N) = 1

0 MOD 3 (base 10)

$M_{3,10}$



0 MOD 3 (base 10)

- **Theorem 1.C** :
  Let $w \in \{0,1,\ldots,9\}^*$ be of length $n \geq 0$.

  1) M_1 stops in state $q_0 \Leftrightarrow w = 0 \bmod 3$.

  2) M_1 stops in state $q_1 \Leftrightarrow w = 1 \bmod 3$.

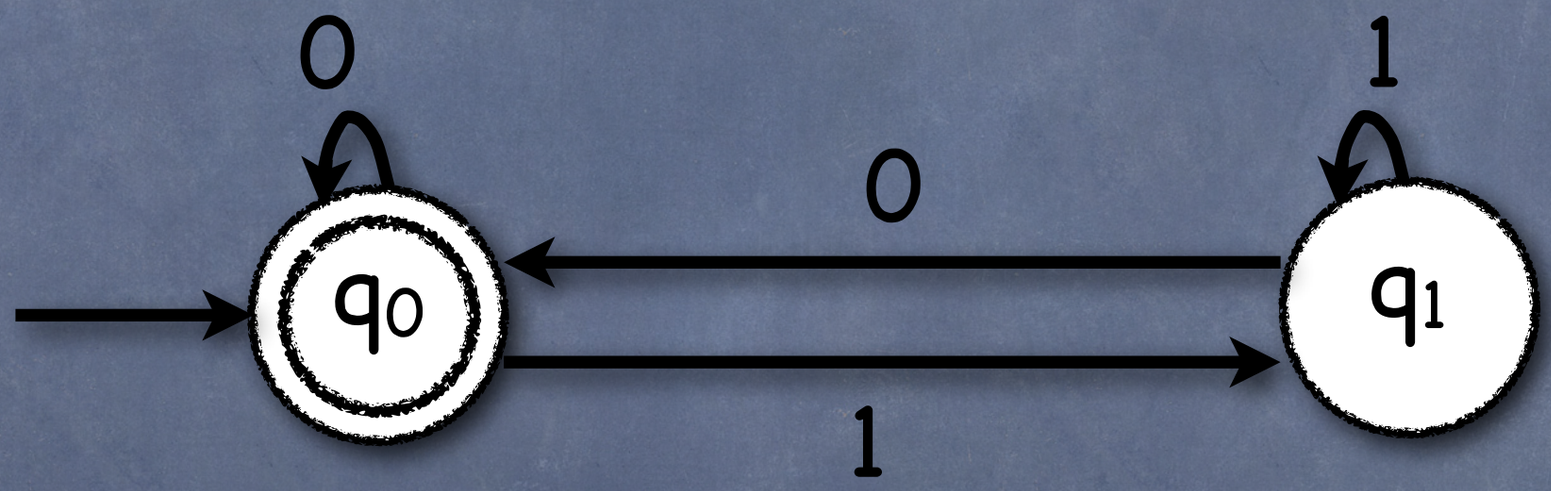
  3) M_1 stops in state $q_2 \Leftrightarrow w = 2 \bmod 3$.

Examples: automata for multiples of N base B

- automata for multiples of N = 0 mod N
- examples mod 2, mod 3, mod 7

0 MOD 2 (base 2)

$M_{2,2}$



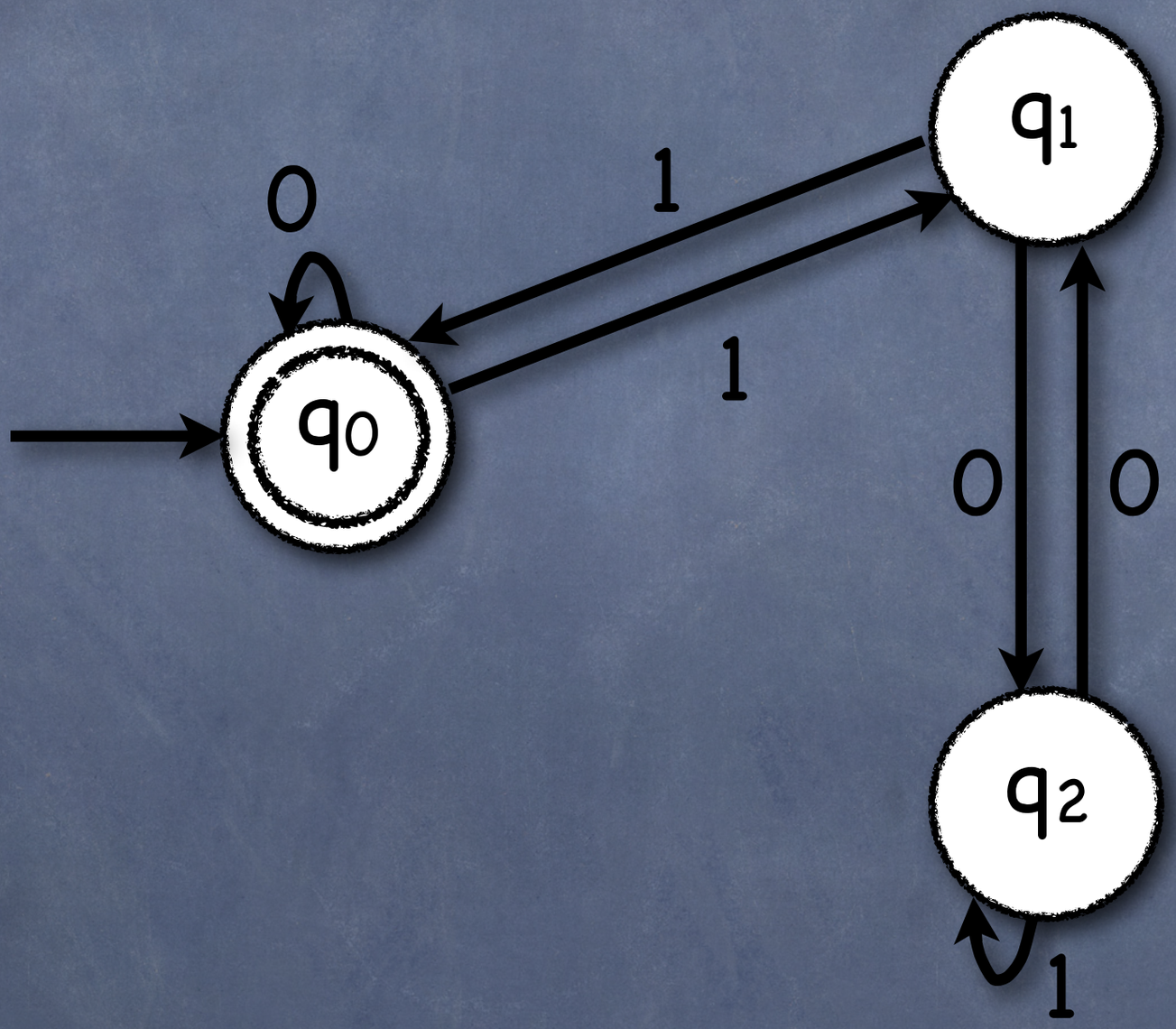
- Remember what you learned in school of CS:
  N (in binary) is a multiple of 2 iff it ends by 0.

$M_{2,2}$ stops in state $q_r \Leftrightarrow w = r \bmod 2$

gcd(B,N) = 1

0 MOD 3 (base 2)

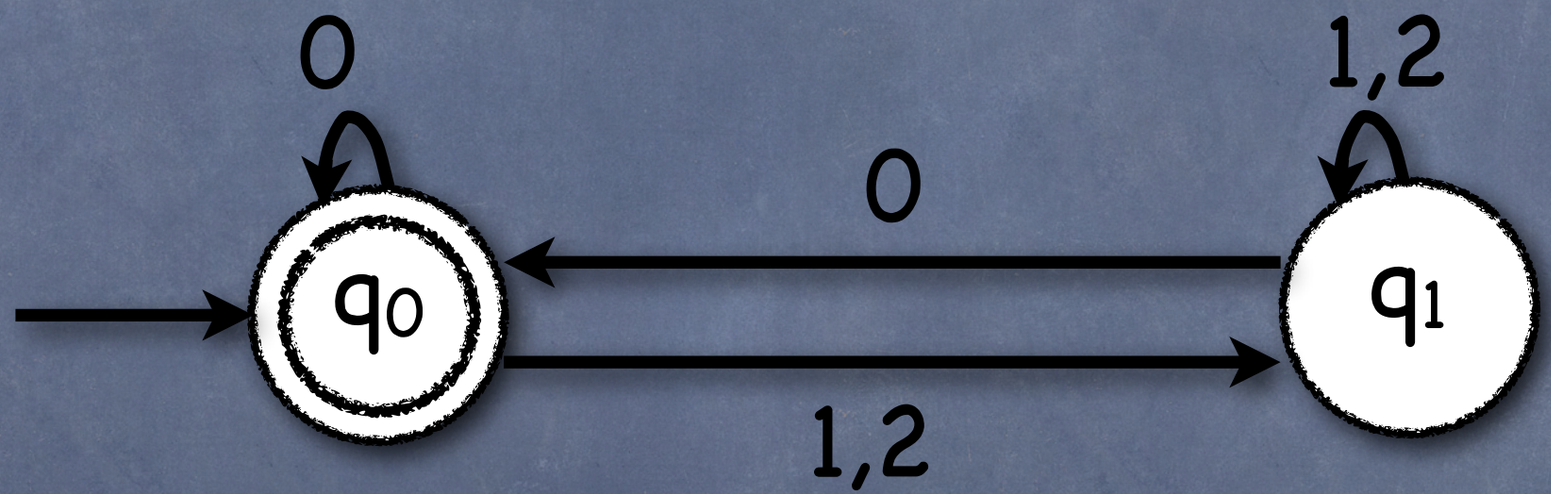
$M_{3,2}$



$M_{3,2}$ stops in state $q_r \Leftrightarrow w = r \bmod 3$

0 MOD 3 (base 3)

$M'_{3,3}$



- Remember what you learned in school of CS:
  N (in ternary) is a multiple of 3 iff it ends by 0.

$M'_{3,3}$ stops in state $q_0 \Leftrightarrow w = 0 \bmod 3$

$M_{7,10}$

multiples of 7

q_0 q_1 q_2 q_3 q_4 q_5 q_6

Another example: multiples of 7...

Another example: multiples of 7...

Another example: multiples of 7...

Another example: multiples of 7...

Another example: multiples of 7...

Another example: multiples of 7...

Another example: multiples of 7...

Another example: multiples of 7...

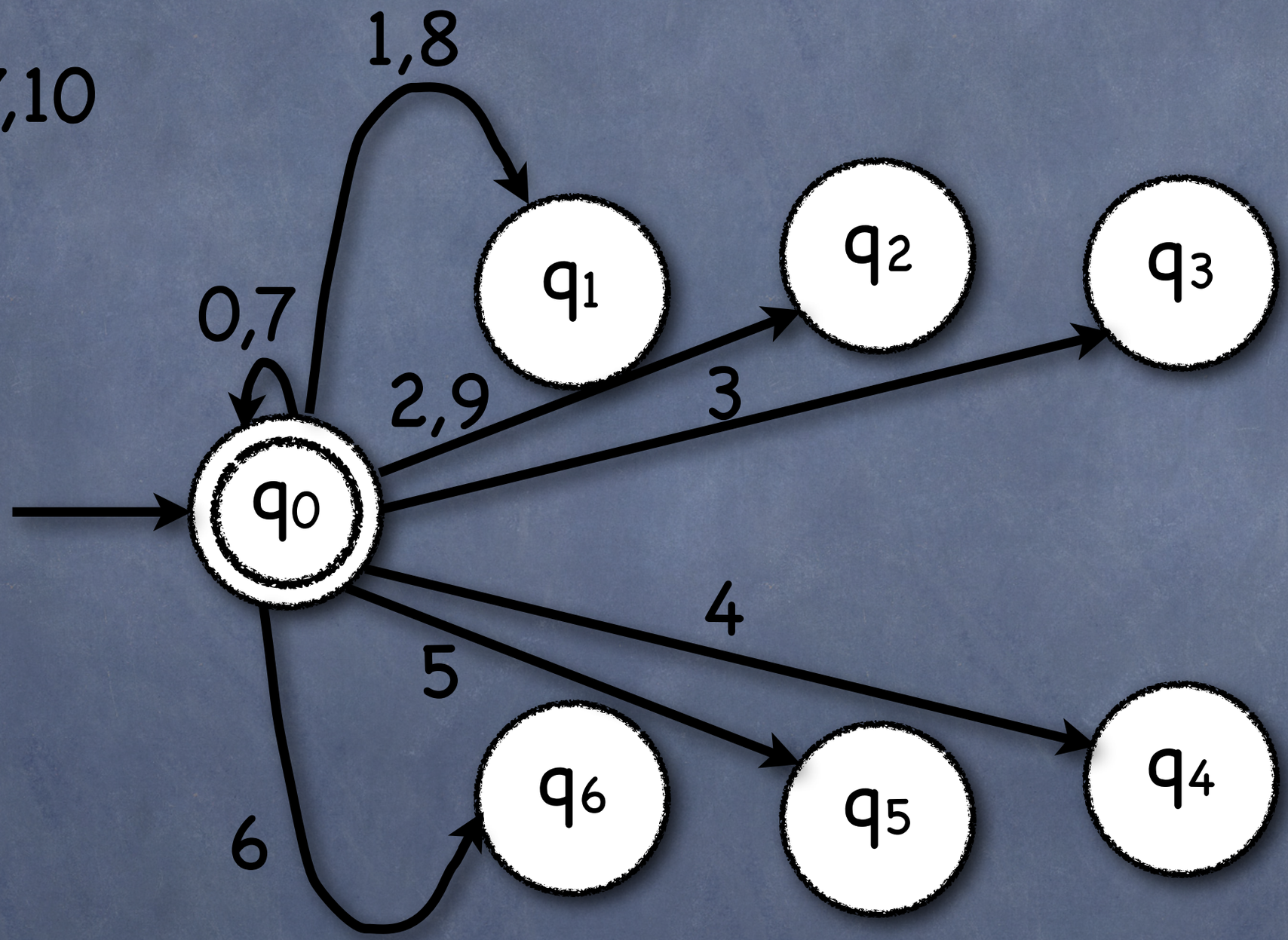
- Remember forever what you learn in COMP-330 today : N is a multiple of 7 if $N \in L(M_{7,10})$.

- Example: 54705 is a multiple of 7 because
- $5 = (10 \times 0 + 5) = 5 = 5 \bmod 7$,
- $54 = (10 \times 5 + 4) = 54 = 5 \bmod 7$,
- $547 = (10 \times 5 + 7) = 57 = 1 \bmod 7$,
- $5470 = (10 \times 1 + 0) = 10 = 3 \bmod 7$ and
- $54705 = (10 \times 3 + 5) = 35 = 0 \bmod 7$.

gcd(B,N) = 1

0 MOD 7 (base 10)

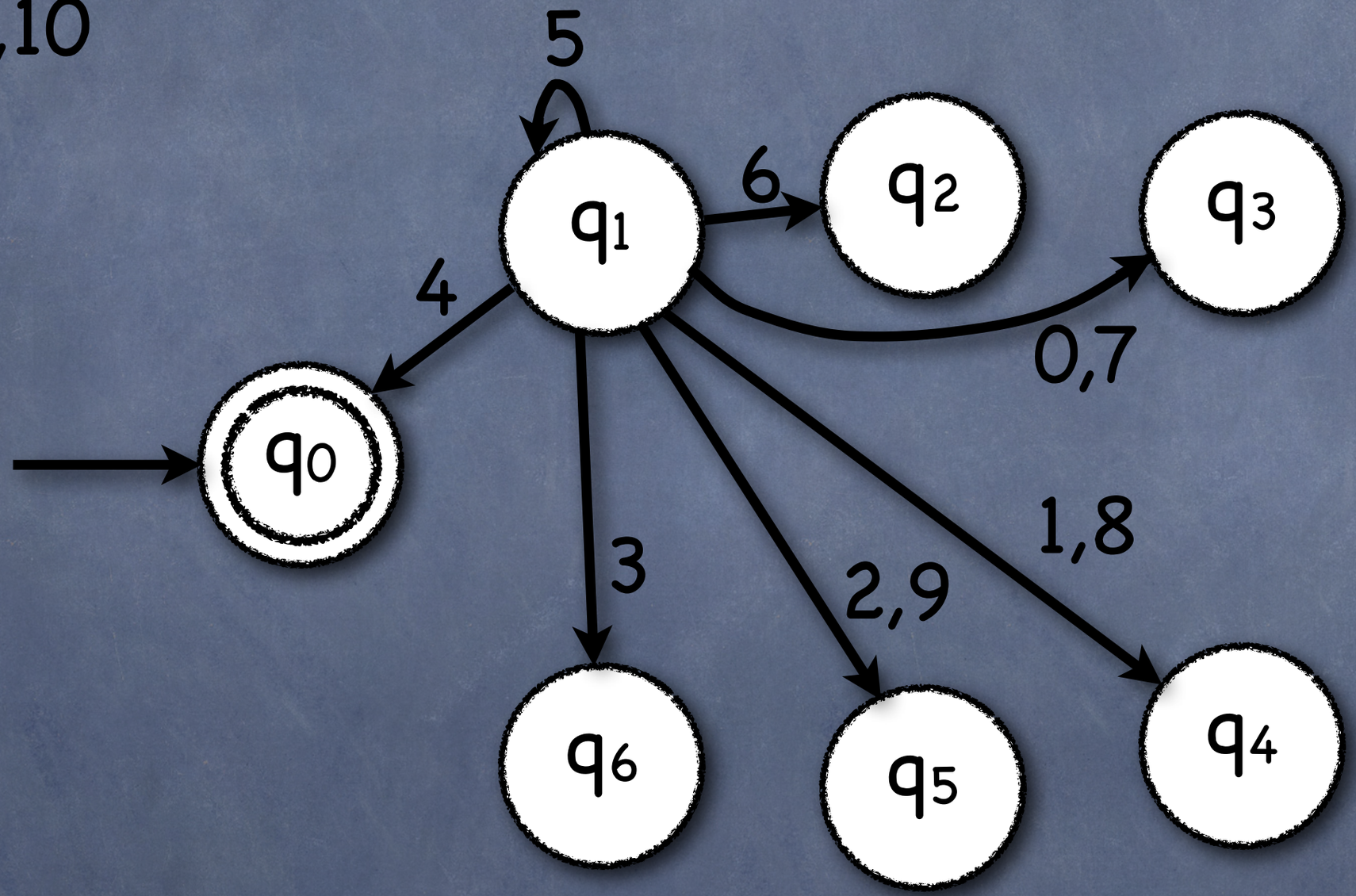
$M^0_{7,10}$



$M_{7,10}$ stops in state $q_r \Leftrightarrow w = r \bmod 7$

1 MOD 7 (base 10)

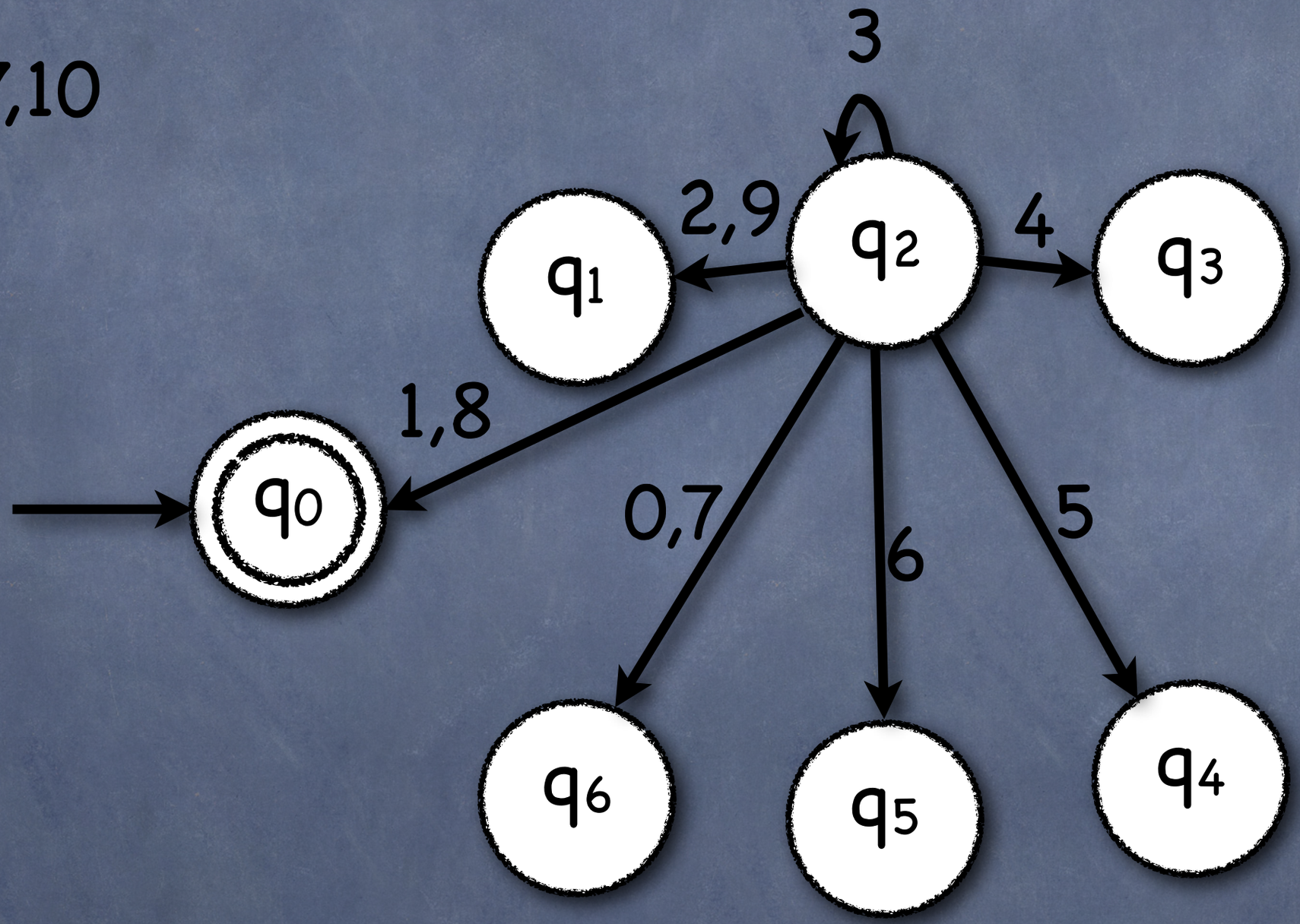
$M^1_{7,10}$



$M_{7,10}$ stops in state $q_r \Leftrightarrow w = r \bmod 7$

2 MOD 7 (base 10)

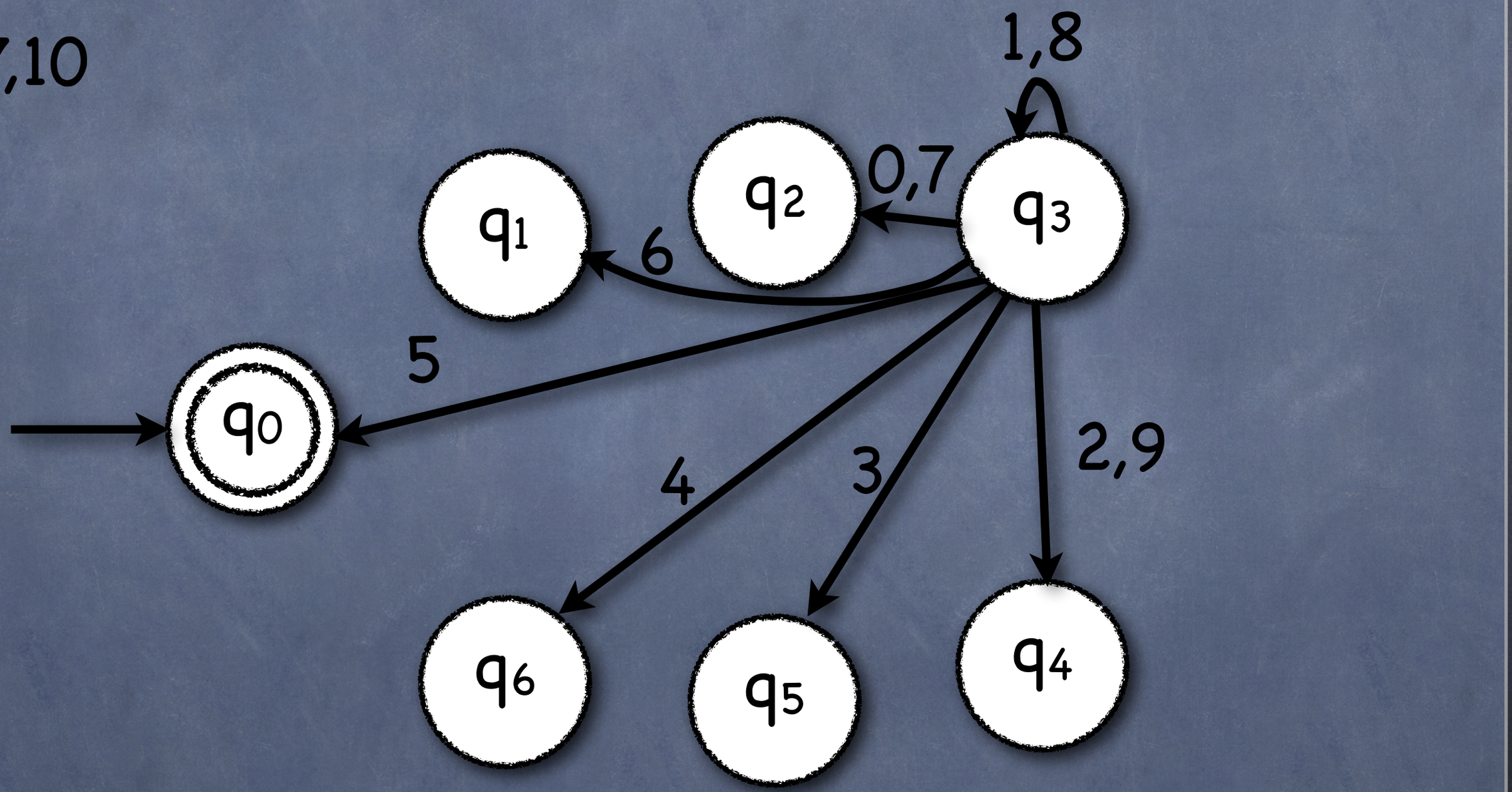
$M^2_{7,10}$



$M_{7,10}$ stops in state $q_r \Leftrightarrow w = r \bmod 7$

3 MOD 7 (base 10)

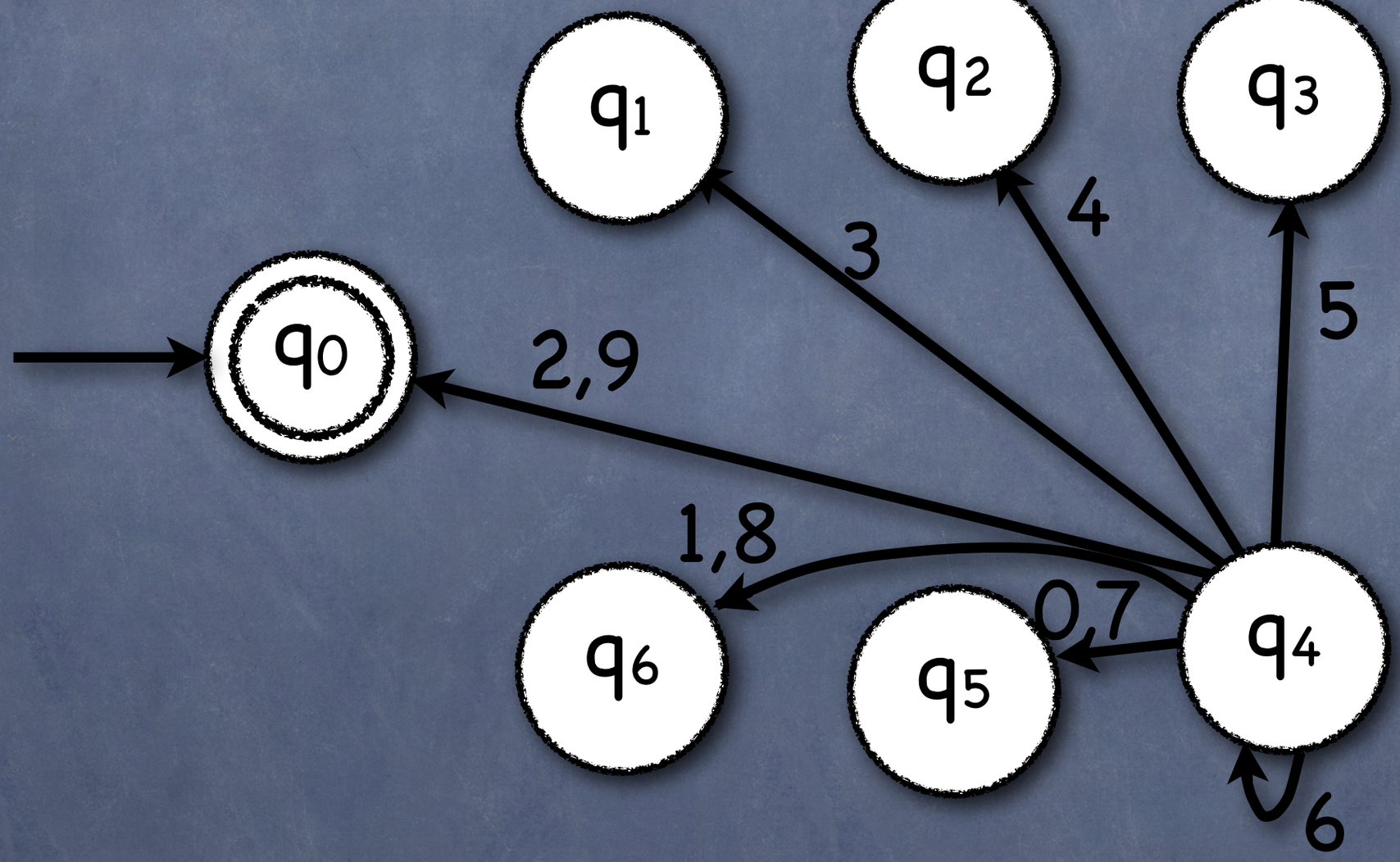
$M^3_{7,10}$



$M_{7,10}$ stops in state $q_r \Leftrightarrow w = r \bmod 7$

4 MOD 7 (base 10)

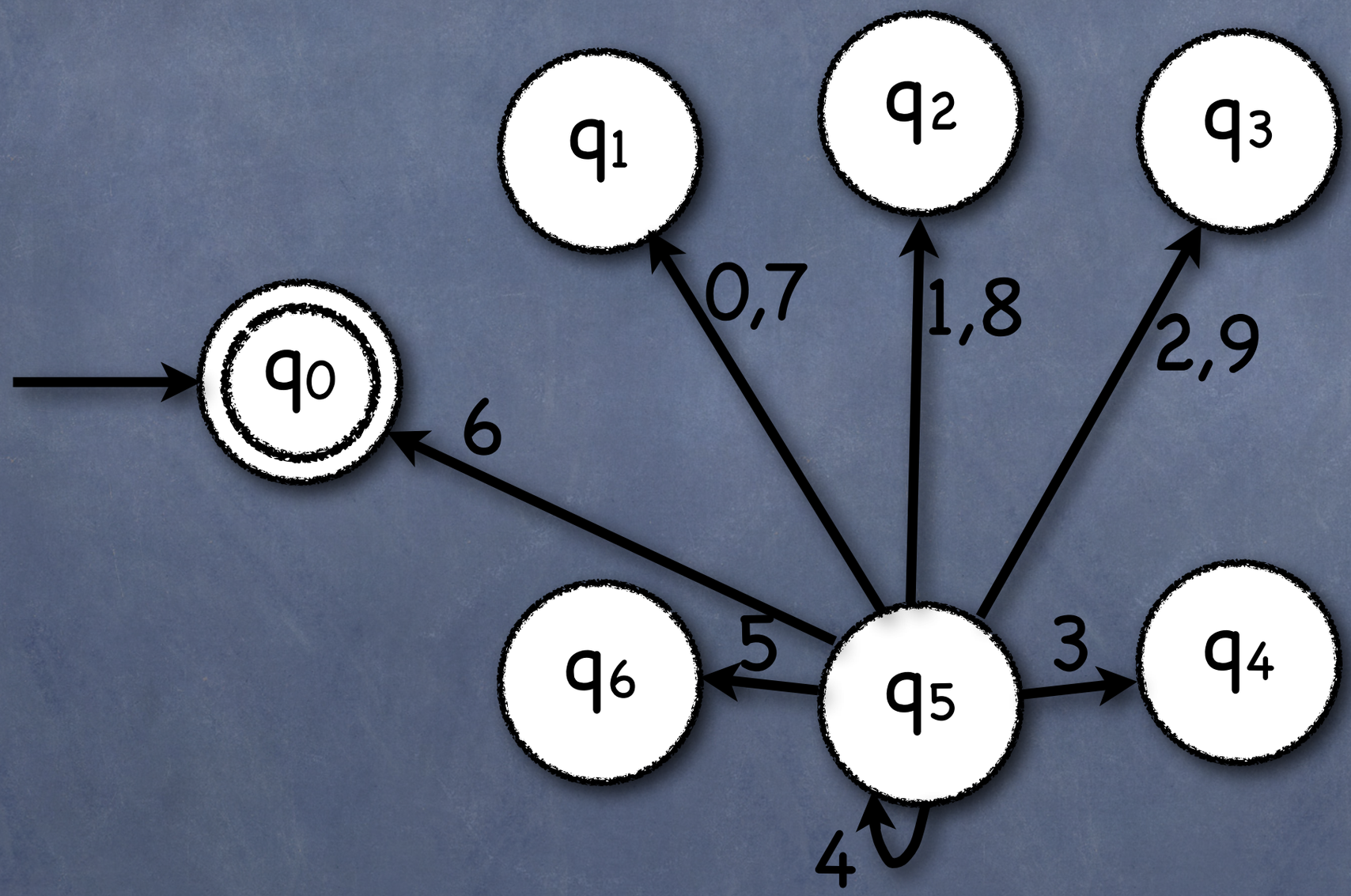
$M^4_{7,10}$



$M_{7,10}$ stops in state $q_r \Leftrightarrow w = r \bmod 7$

5 MOD 7 (base 10)

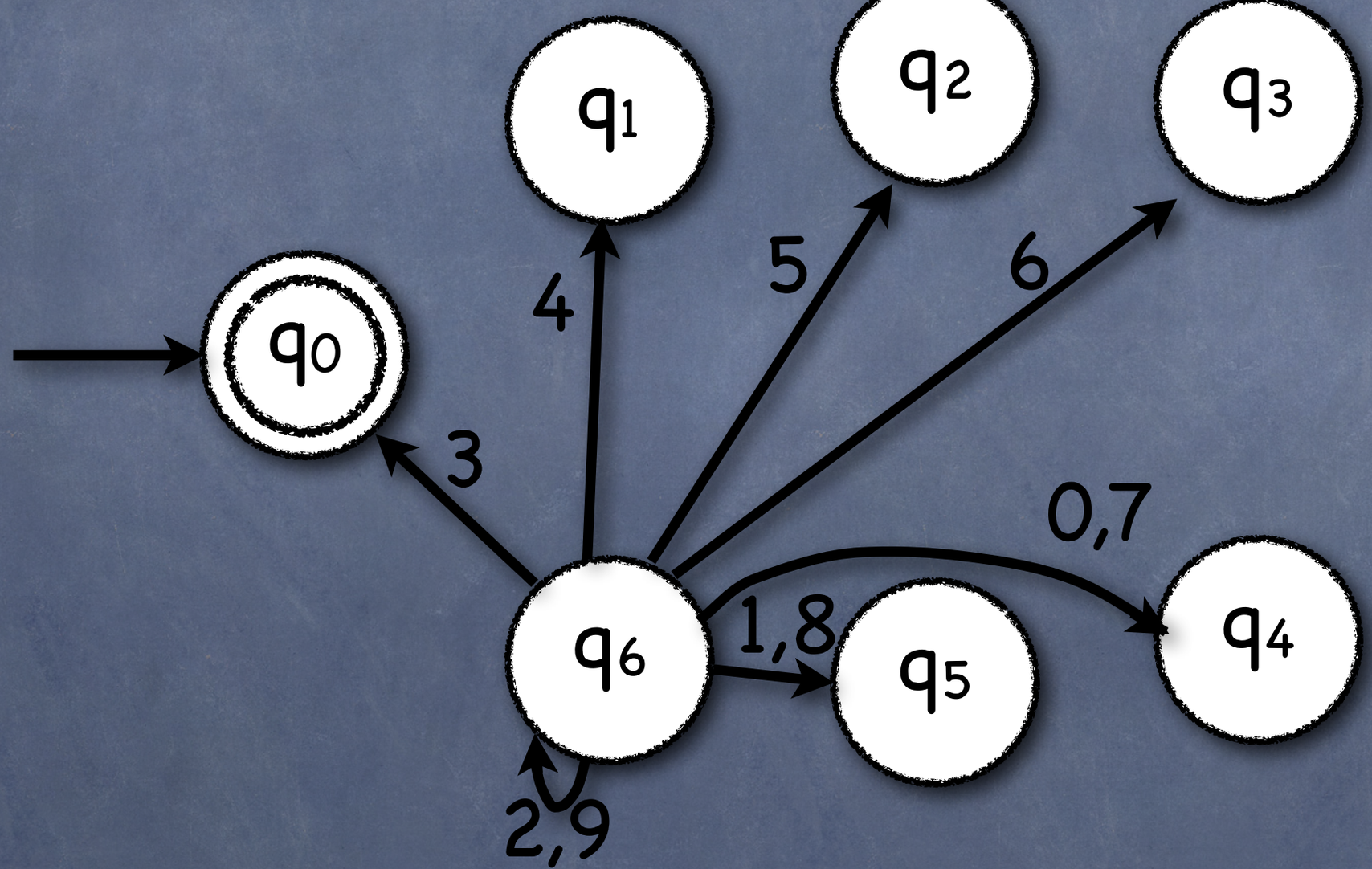
$M^5_{7,10}$



$M_{7,10}$ stops in state $q_r \Leftrightarrow w = r \bmod 7$

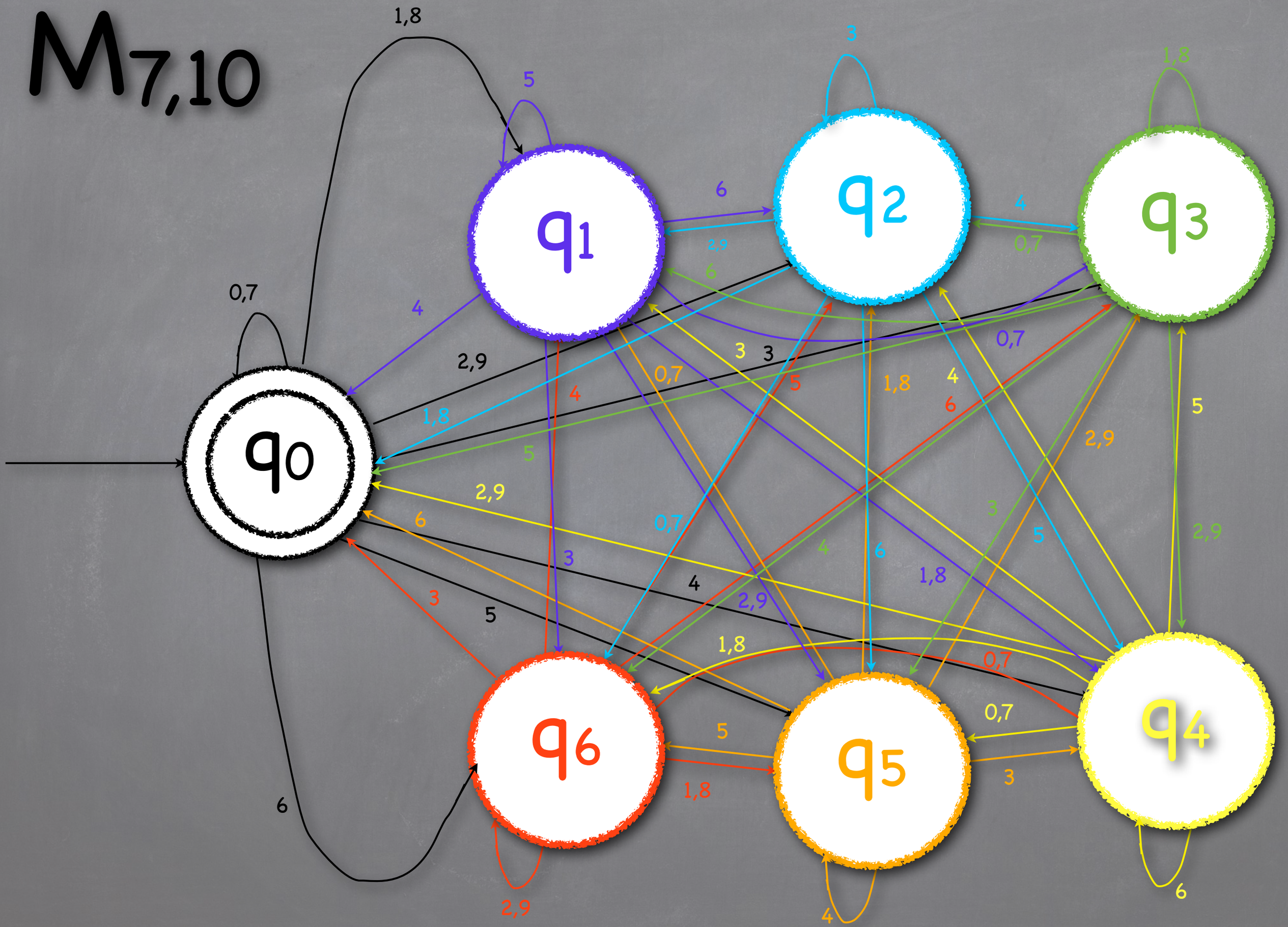
6 MOD 7 (base 10)

$M^6_{7,10}$



$M_{7,10}$ stops in state $q_r \Leftrightarrow w = r \bmod 7$

$M_{7,10}$


q0
q1
q2
q3
q4
q5
q6
1,8
5
3
1,8
0,7
6
4
2,9
6
4
0,7
2,9
3 3
0,7
4
5
1,8
4
6
0,7
5
2,9
6
3
0,7
3
5
4
6
1,8
2,9
1,8
0,7
5
1,8
0,7
3
6
2,9
4
2,9
3
5
6

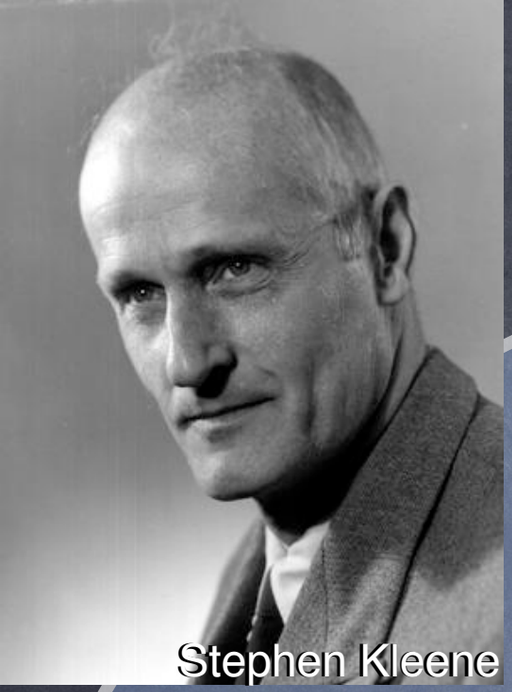
Regular Operations

Regular Operations

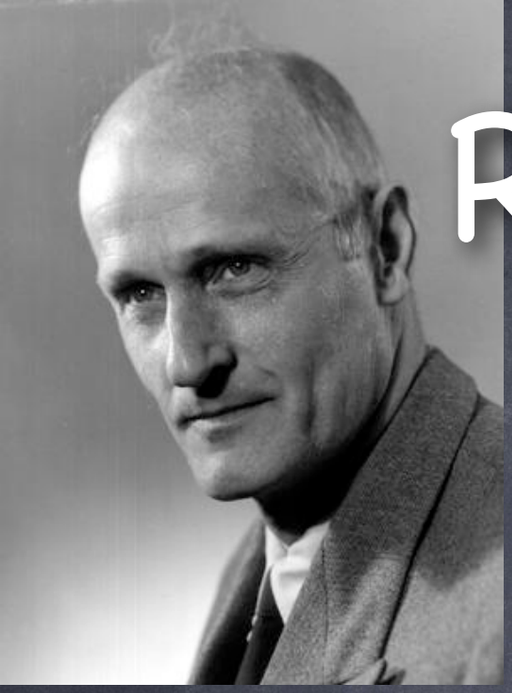
DEFINITION 1.23

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


Stephen Kleene

Regular Operations : Kleene's theorem



Regular Operations : Kleene's theorem

THEOREM 1.25

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Regular Operations : Kleene's theorem

Regular Operations : Kleene's theorem

Regular Operations : Kleene's theorem

Regular Operations : Kleene's theorem

- Let $M_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a DFA accepting L_A and $M_B=(Q_B,\Sigma,\delta_B,q_{0B},F_B)$ be a DFA accepting L_B.

- Consider $M_U=(Q_A \times Q_B,\Sigma,\delta_U,(q_{0A},q_{0B}),F_U)$ where

  $\delta_U((q,q'),s) = (\delta_A(q,s), \delta_B(q',s))$ for all q,q',s

  and

  $F_U = \{ (q,q') \mid q \in F_A \text{ or } q' \in F_B \}$.

- $L_U = L_A \cup L_B$.

- Let $M_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a DFA accepting L_A and $M_B=(Q_B,\Sigma,\delta_B,q_{0B},F_B)$ be a DFA accepting L_B.

- Consider $M_U=(Q_A \times Q_B,\Sigma,\delta_U,(q_{0A},q_{0B}),F_U)$ where

  $\delta_U((q,q'),s) = (\delta_A(q,s), \delta_B(q',s))$ for all q,q',s

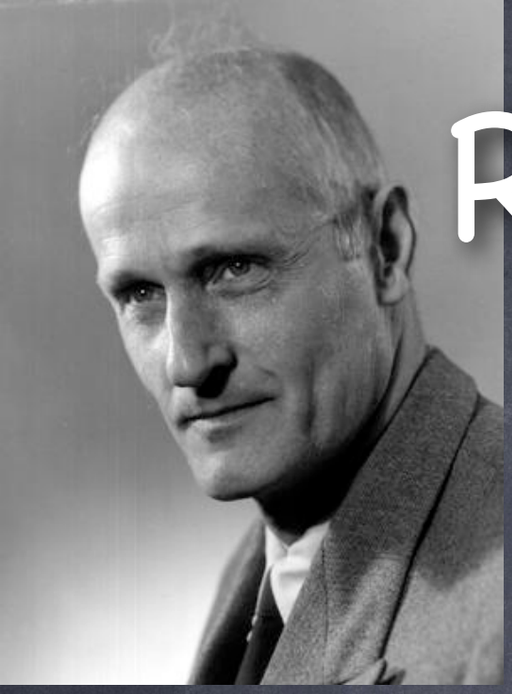
  and

  $F_U = \{ (q,q') \mid q \in F_A \text{ or } q' \in F_B \}$.

- $L_U = L_A \cup L_B$.

- We can write it as $F_U = (F_A \times Q_B) \cup (Q_A \times F_B)$.
  (Not the same as $F_A \times F_B$.)

- The resulting language would be the **intersection** and not the **union**. This proves that the class of regular languages is closed under **intersection**.



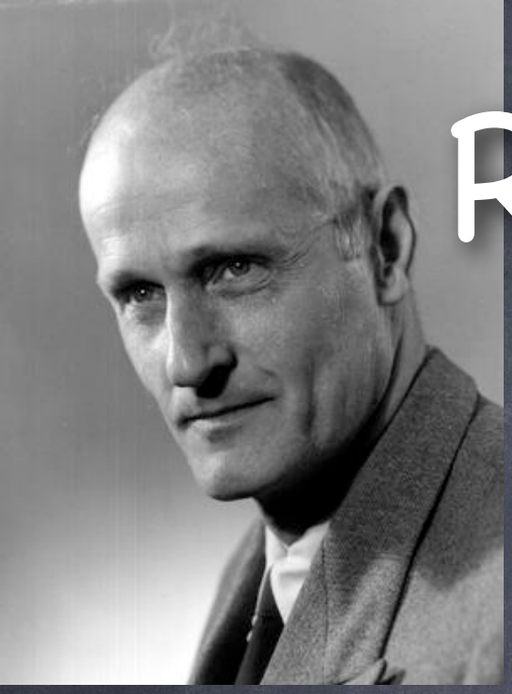
Regular Operations : Kleene's theorem (DFA)

THEOREM 1.26

The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

(NFA)



Regular Operations : Kleene's theorem (DFA)

THEOREM 1.26

The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

(NFA)

THEOREM 1.47

The class of regular languages is closed under the concatenation operation.

Regular Operations : Kleene's theorem (NFA)

Regular Operations : Kleene's theorem (NFA)

THEOREM 1.45

The class of regular languages is closed under the union operation.

Regular Operations : Kleene's theorem (NFA)

THEOREM 1.47

The class of regular languages is closed under the concatenation operation.

THEOREM 1.45

The class of regular languages is closed under the union operation.

Regular Operations : Kleene's theorem (NFA)

THEOREM 1.49

The class of regular languages is closed under the star operation.

THEOREM 1.47

The class of regular languages is closed under the concatenation operation.

THEOREM 1.45

The class of regular languages is closed under the union operation.

Non-Deterministic Finite Automata

Non-Deterministic Finite Automata

N_1

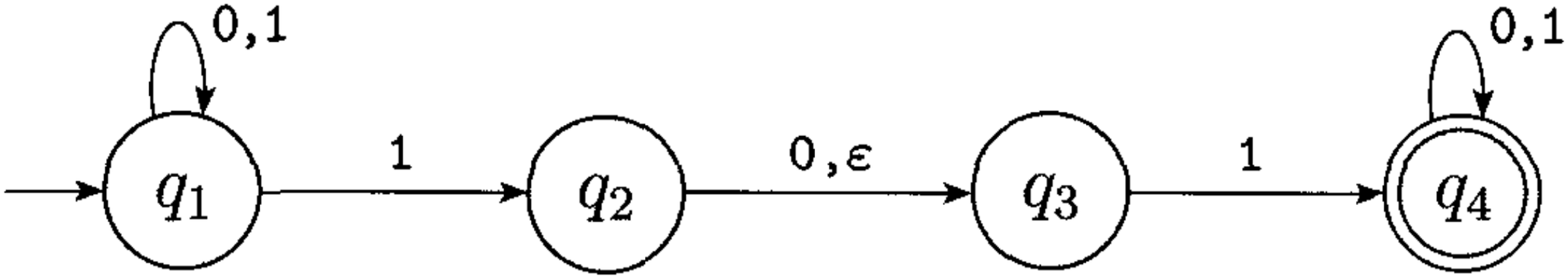


FIGURE 1.27
The nondeterministic finite automaton N_1

Definition of NFA

DEFINITION 1.37

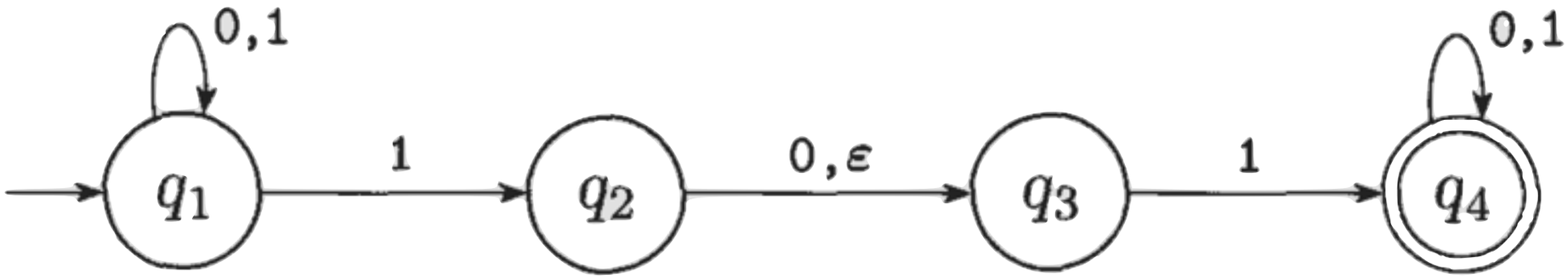
A ***nondeterministic finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta\colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ $\quad$ $\mathcal{P}(Q) = \{ S : S \subseteq Q \}$

EXAMPLE 1.38

Recall the NFA N_1:



The formal description of N_1 is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0,1\}$,
3. δ is given as

	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	$\{q_4\}$	$\{q_4\}$	$\emptyset$

,

4. q_1 is the start state, and
5. $F = \{q_4\}$.

Definition of NFA

- Let $N = (Q,\Sigma,\delta,q_0,F)$ be a nondeterministic finite state automaton and let $w=w_1w_2...w_n$ ($n\geq 0$) be a string where each symbol $w_i \in \Sigma$.

- N **accepts** w if $\exists\ m\geq n$, $\exists\ s_0,s_1,...,s_m$ and $\exists\ y_1y_2...y_m = w$, with each $y_i \in \Sigma_\varepsilon$ s.t.
  1. $s_0 = q_0$
  2. $s_{i+1} \in \delta(s_i,y_{i+1})$ for $i = 0 \ldots m-1$, and
  3. $s_m \in F$

NFA-DFA equivalence

Regular Languages

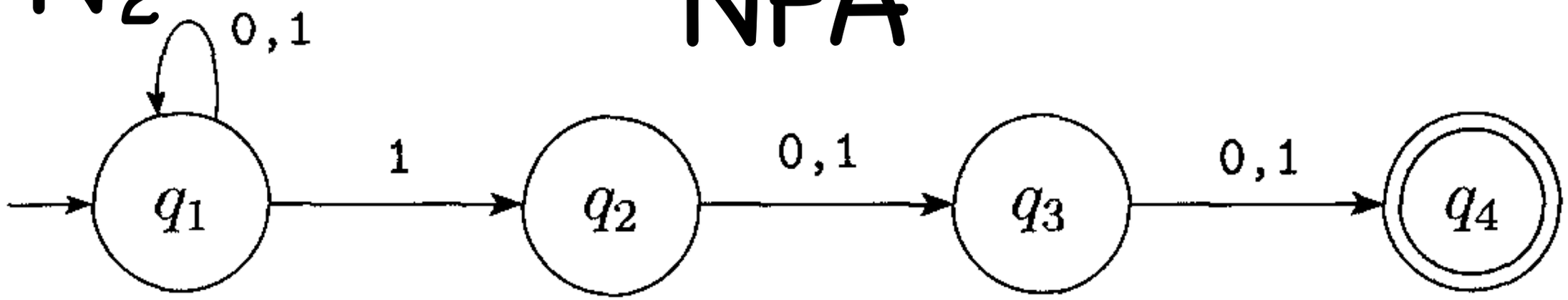
THEOREM 1.39

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

COROLLARY 1.40

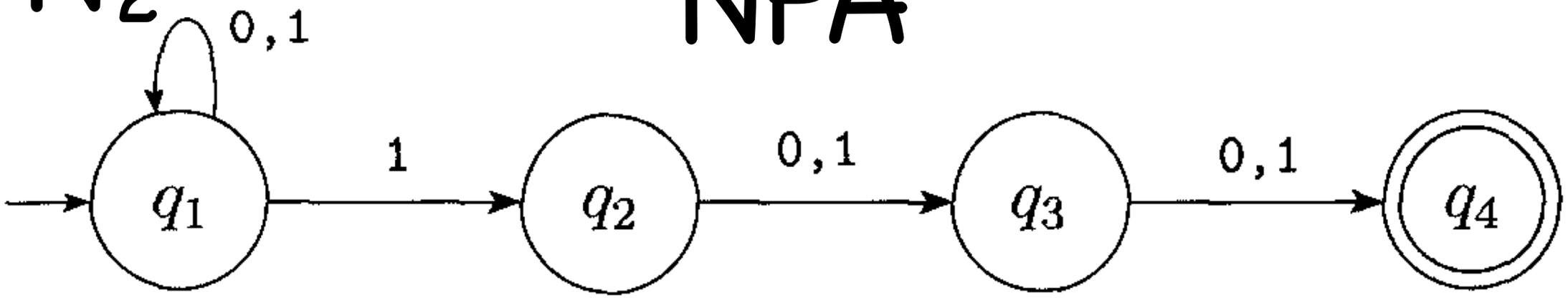
A language is regular if and only if some nondeterministic finite automaton recognizes it.

N_2 NFA



N_2

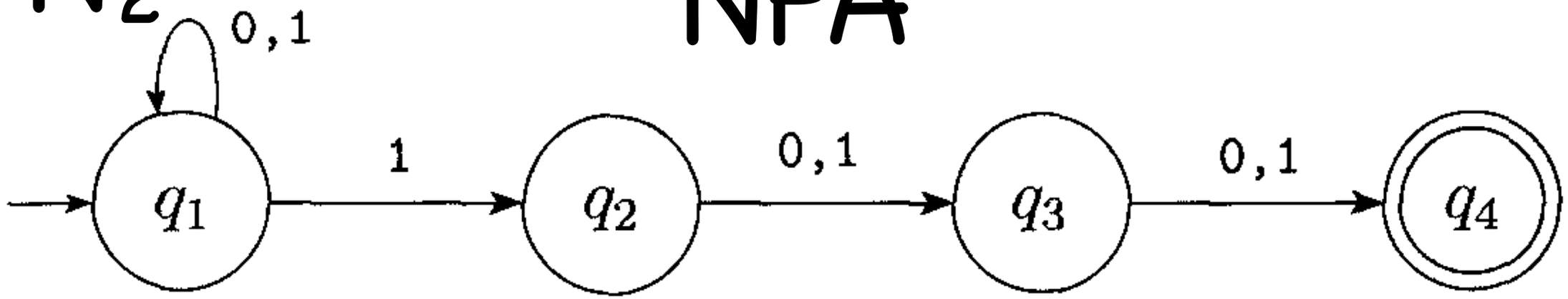
NFA



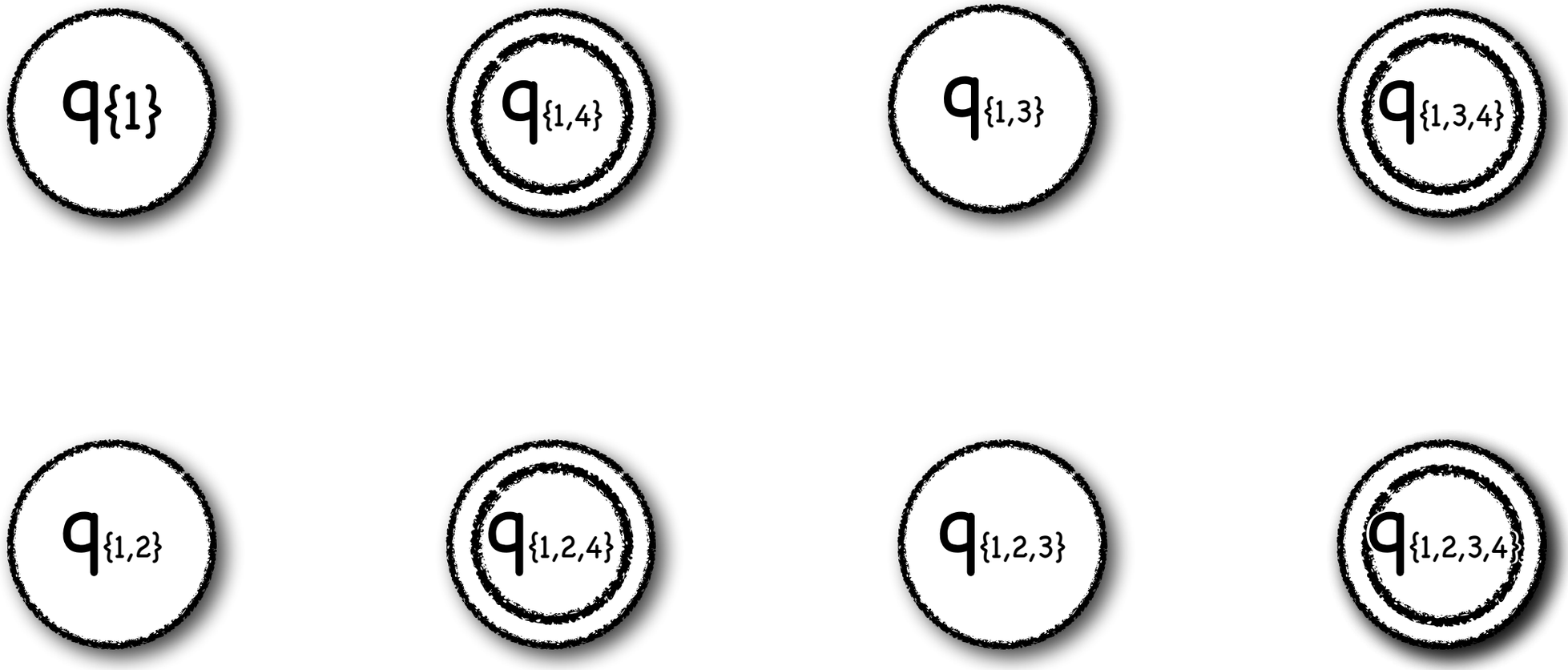
DFA

N_2

NFA

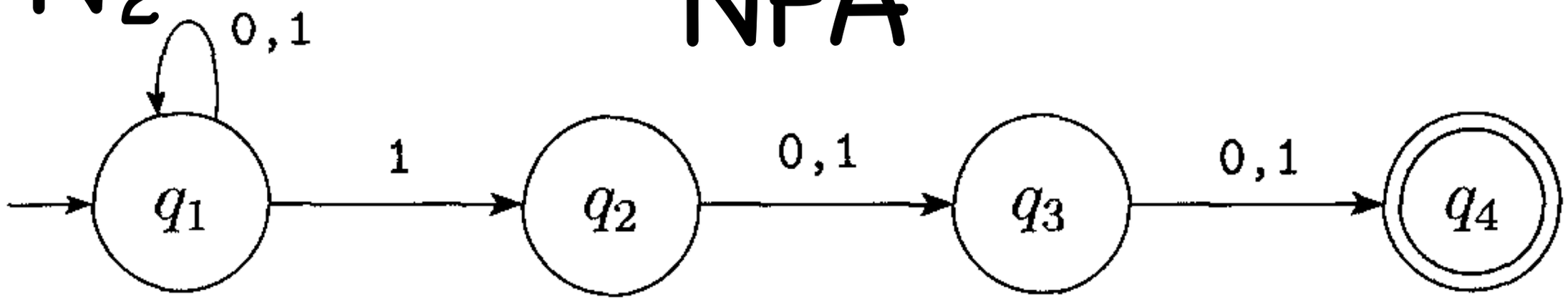


DFA

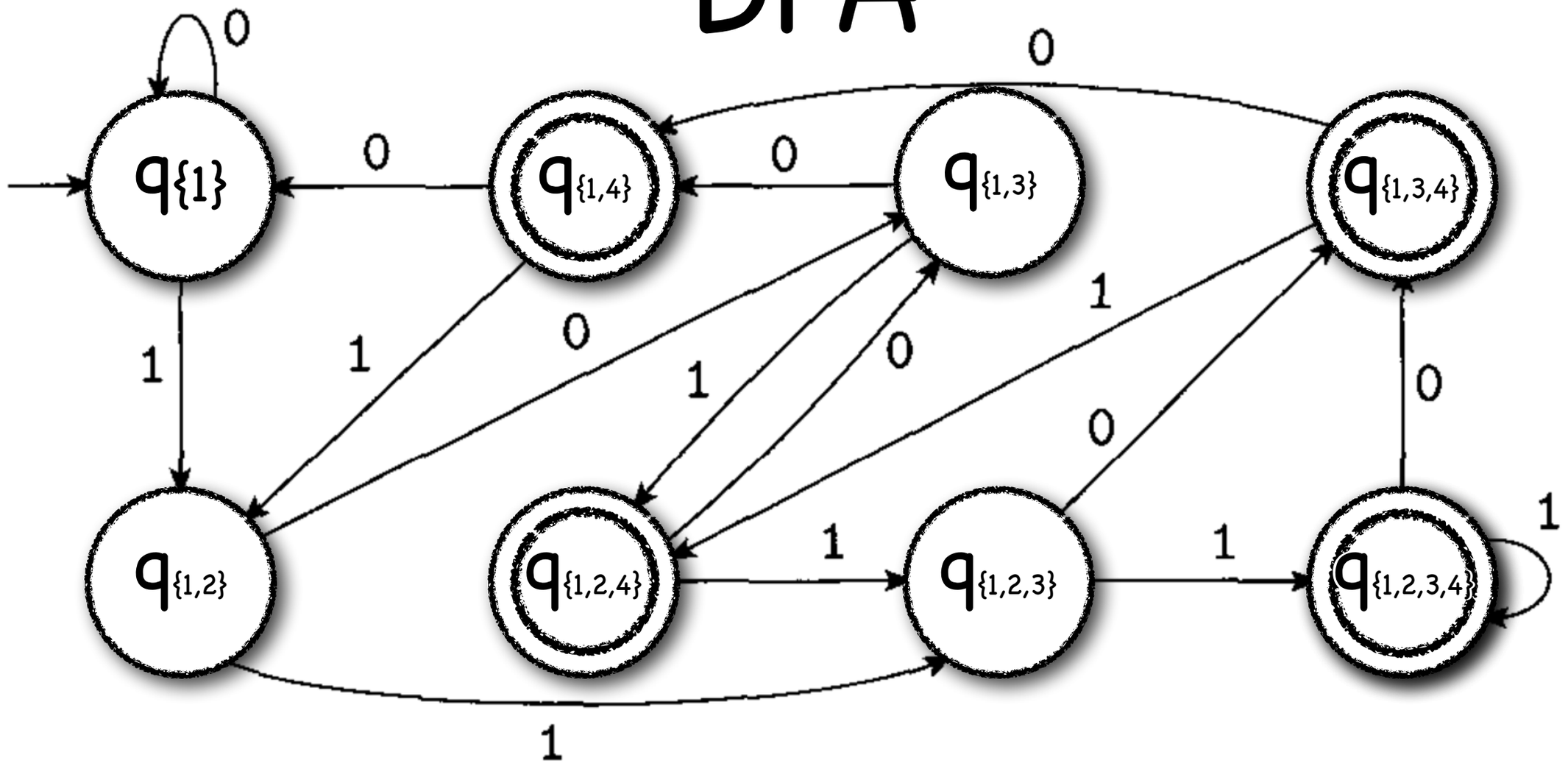


N_2

NFA


0,1
1
0,1
0,1
q_1
q_2
q_3
q_4

DFA


0
0
0
0
1
1
1
0
1
0
0
1
1
1
1
$q_{\{1\}}$
$q_{\{1,4\}}$
$q_{\{1,3\}}$
$q_{\{1,3,4\}}$
$q_{\{1,2\}}$
$q_{\{1,2,4\}}$
$q_{\{1,2,3\}}$
$q_{\{1,2,3,4\}}$

NFA-DFA equivalence

(without empty transitions)

NFA-DFA equivalence

(without empty transitions)

NFA-DFA equivalence

(without empty transitions)

NFA-DFA equivalence

(without empty transitions)

NFA-DFA equivalence

(without empty transitions)

NFA-DFA equivalence

(without empty transitions)

- Let $N = (Q,\Sigma,\delta,q_0,F)$ be an NFA (without empty transitions) accepting language A. We show a DFA $M = (Q',\Sigma,\delta',q_0',F')$ accepting A.
- $Q' = \mathcal{P}(Q) = \{ R \mid R \subseteq Q \}$
- $\delta'(R,a) = \{ q \in Q \mid \exists r \in R,\ q \in \delta(r,a) \}$
- $q_0' = \{q_0\}$
- $F' = \{ R \in Q' \mid R \cap F \neq \emptyset \}$

NFA-DFA equivalence

(with empty transitions)

$E(R) = \{q|\ q$ can be reached from R by traveling along 0 or more ε arrows$\}$.

NFA-DFA equivalence

(with empty transitions)

NFA-DFA equivalence

(with empty transitions)

NFA-DFA equivalence

(with empty transitions)

NFA-DFA equivalence

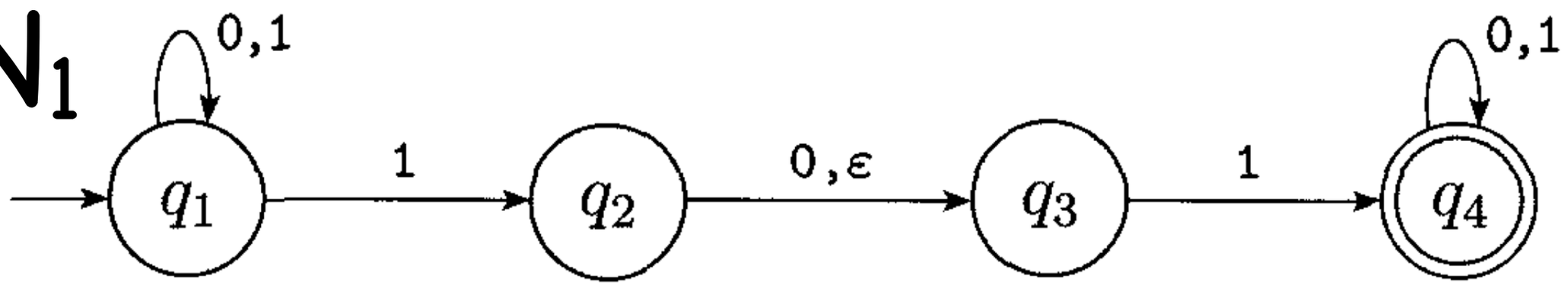
(with empty transitions)

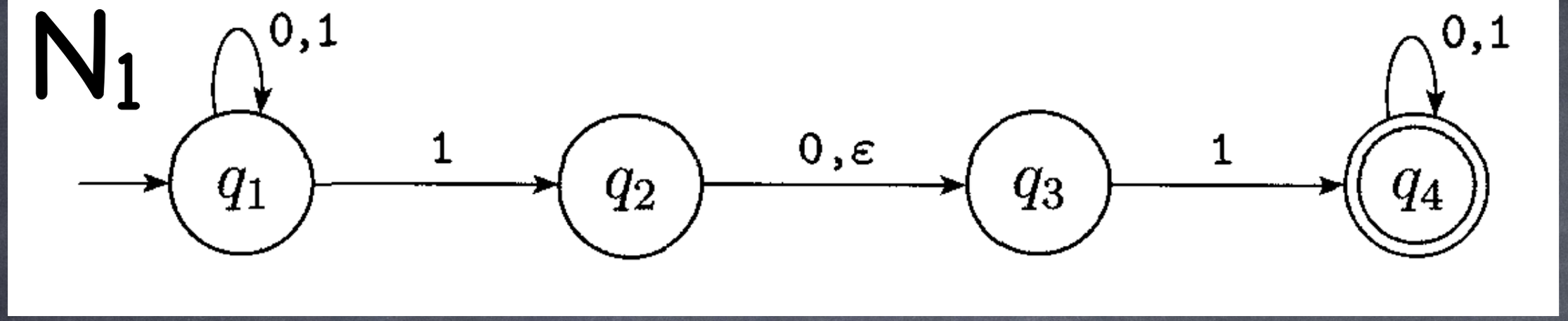
NFA-DFA equivalence

(with empty transitions)

- Let $N = (Q,\Sigma,\delta,q_0,F)$ be an NFA accepting language A. We construct a DFA $M = (Q',\Sigma,\delta',q_0',F')$ accepting A as well.

N_1

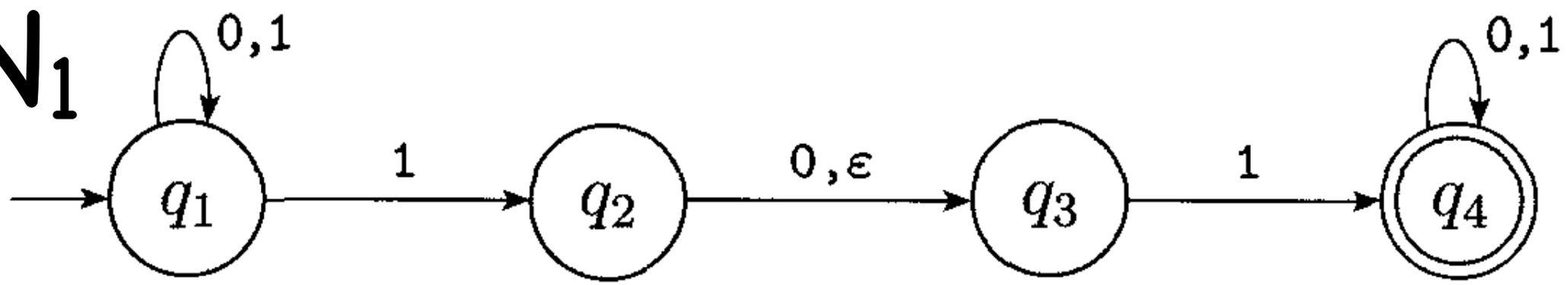



N1
0,1
1
0,ε
1
0,1
q1
q2
q3
q4

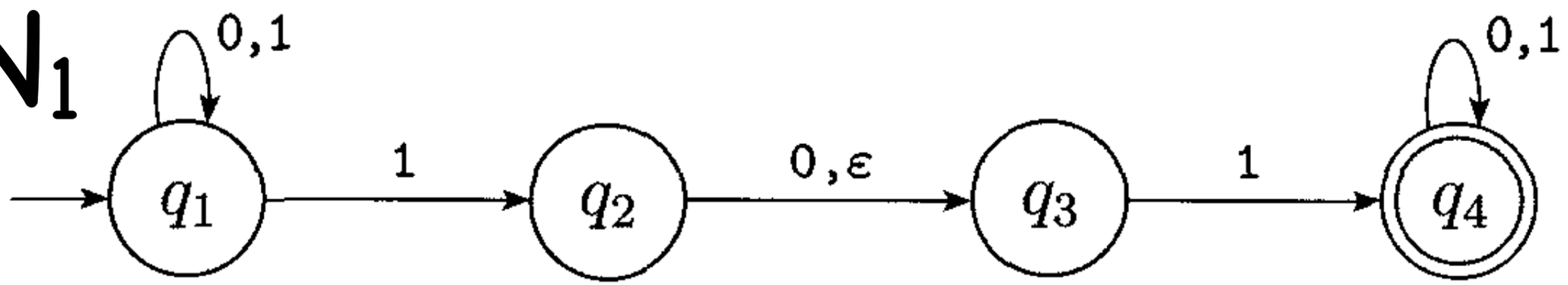

q∅


q{1,2,3,4}

N_1



N_1



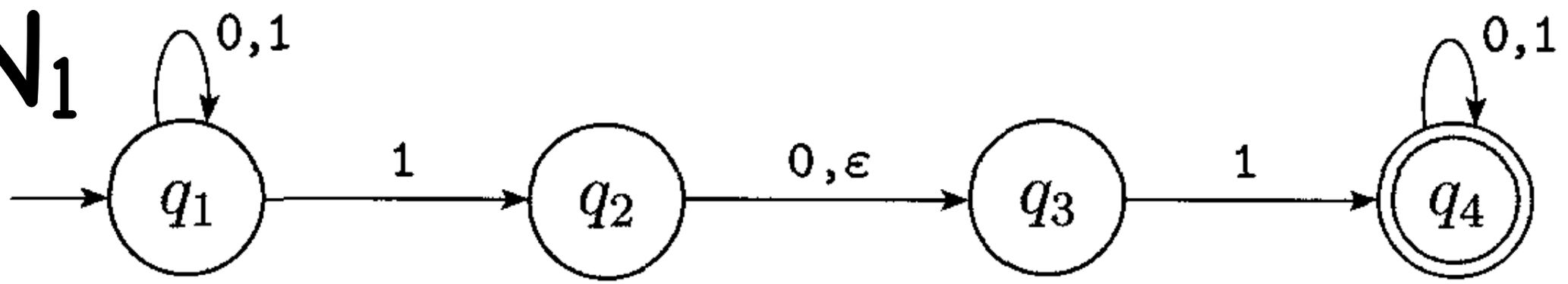
q∅

q{1}

q{2}

q{1,2,3,4}

N_1



q∅

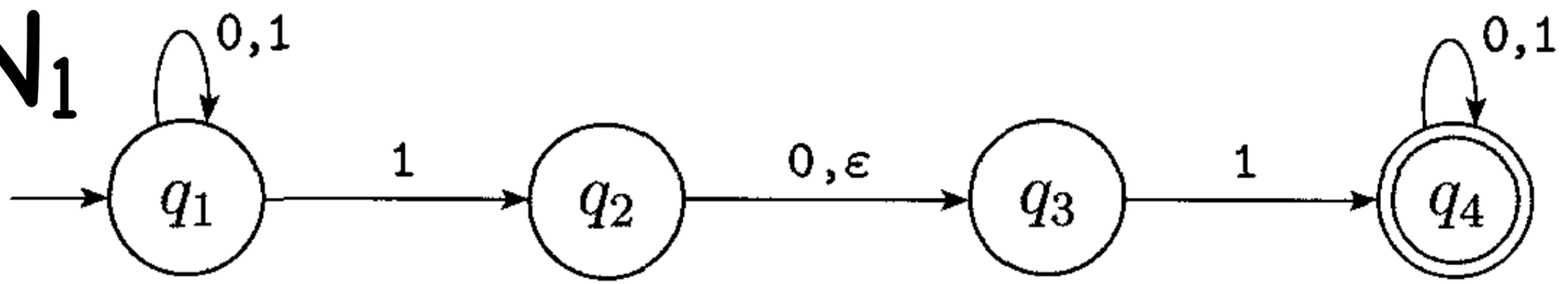
q{1}

q{2}

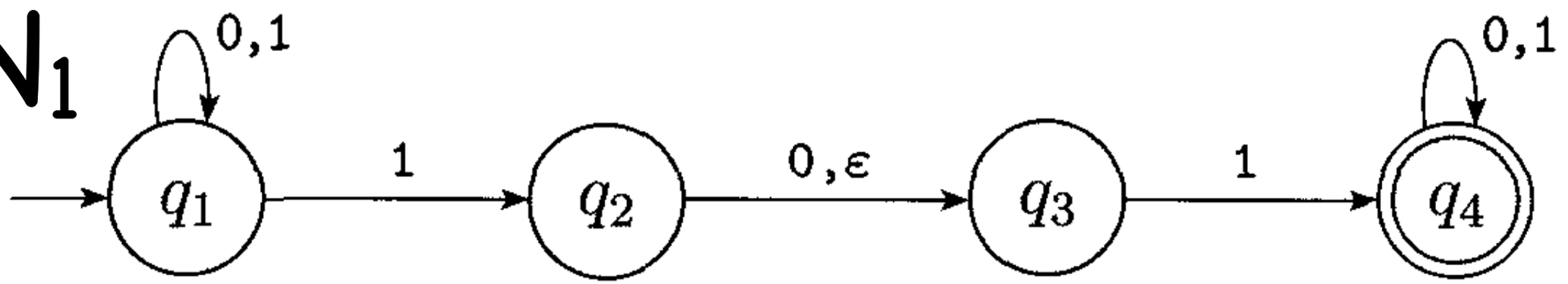
q{3}

q{1,2,3,4}

N_1



N_1



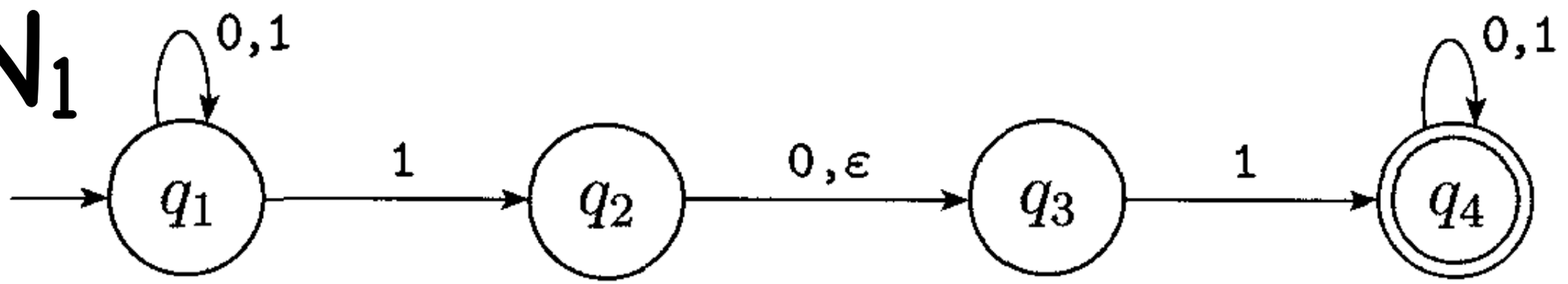
$q_\varnothing$ $q_{\{1\}}$

$q_{\{1,2\}}$ $q_{\{2\}}$

$q_{\{3\}}$

$q_{\{4\}}$ $q_{\{1,2,3,4\}}$

N_1



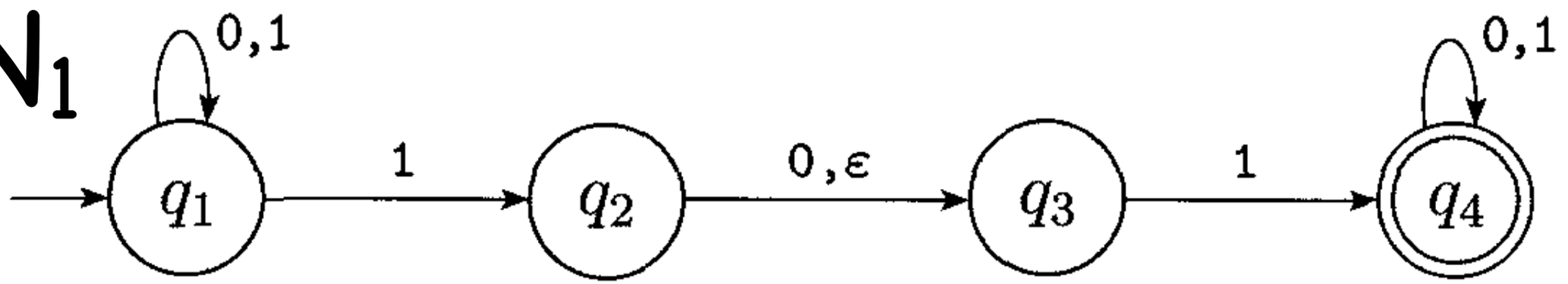
$q_\varnothing$ $q_{\{1\}}$

$q_{\{1,2\}}$ $q_{\{2\}}$

$q_{\{1,3\}}$ $q_{\{3\}}$

$q_{\{4\}}$ $q_{\{1,2,3,4\}}$

N_1



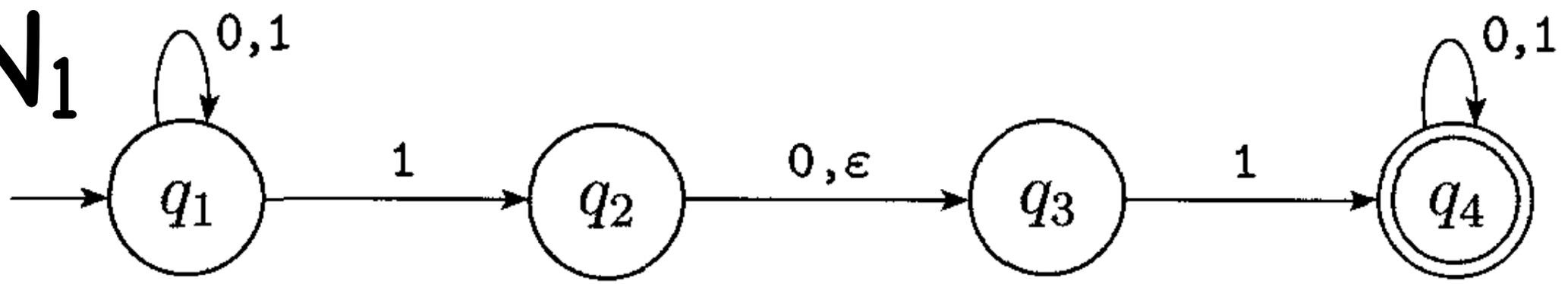
$q_\varnothing$ $q_{\{1\}}$

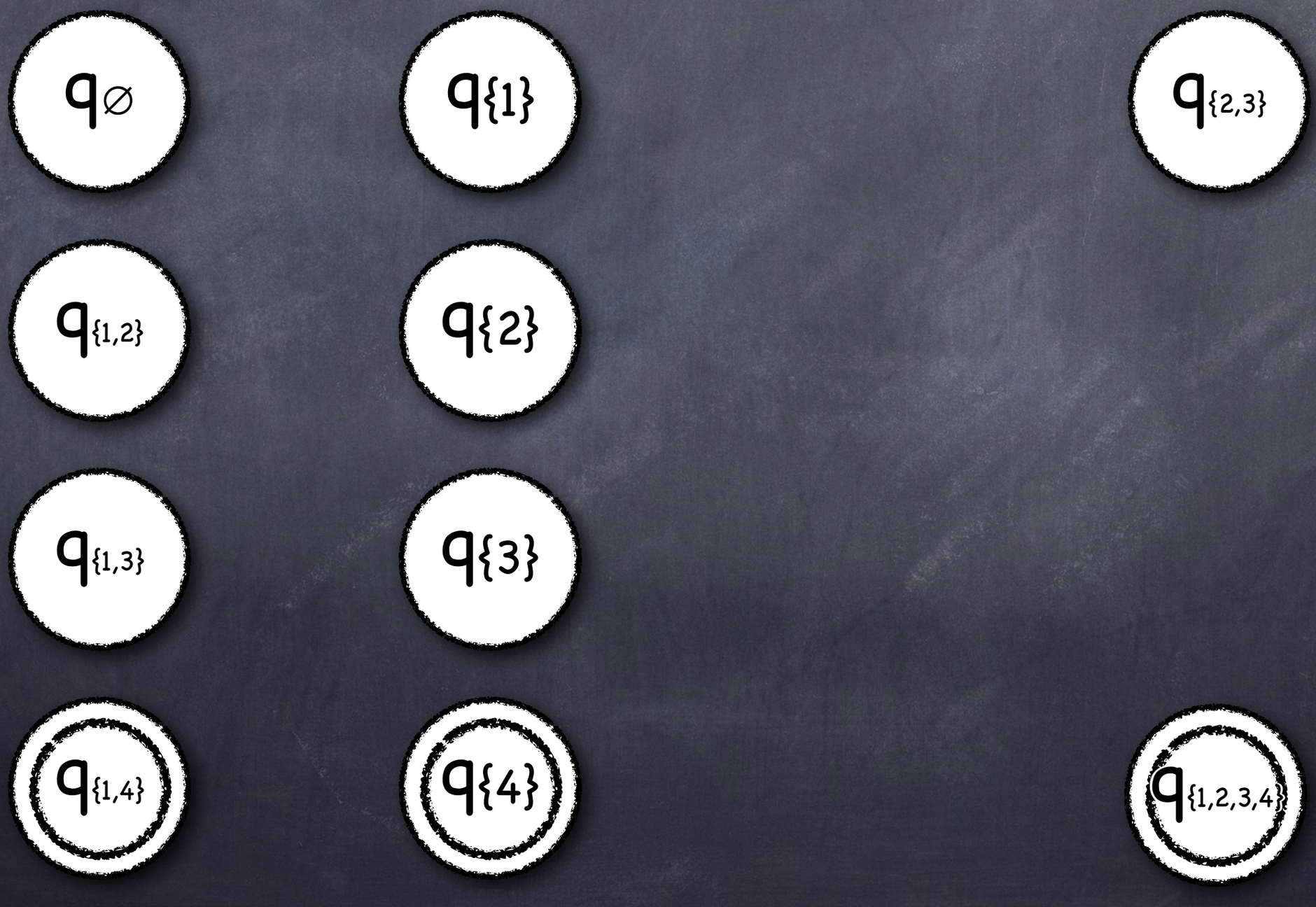
$q_{\{1,2\}}$ $q_{\{2\}}$

$q_{\{1,3\}}$ $q_{\{3\}}$

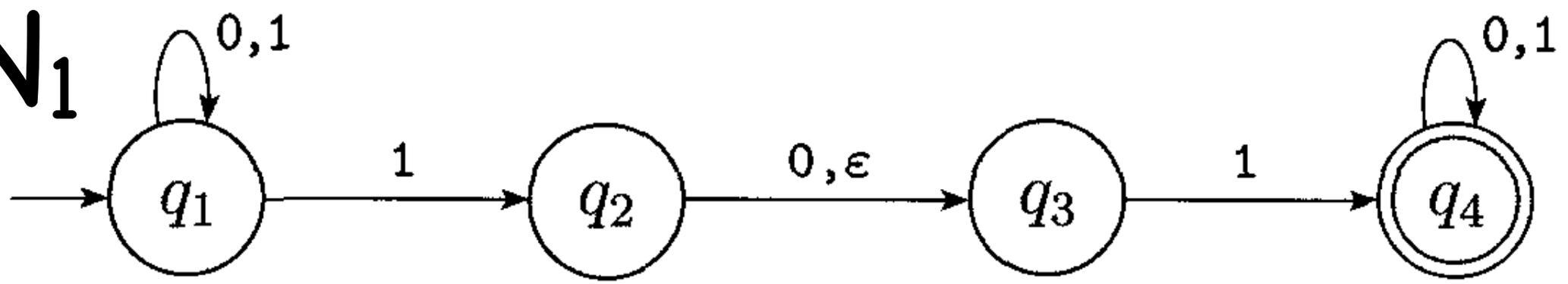
$q_{\{1,4\}}$ $q_{\{4\}}$ $q_{\{1,2,3,4\}}$

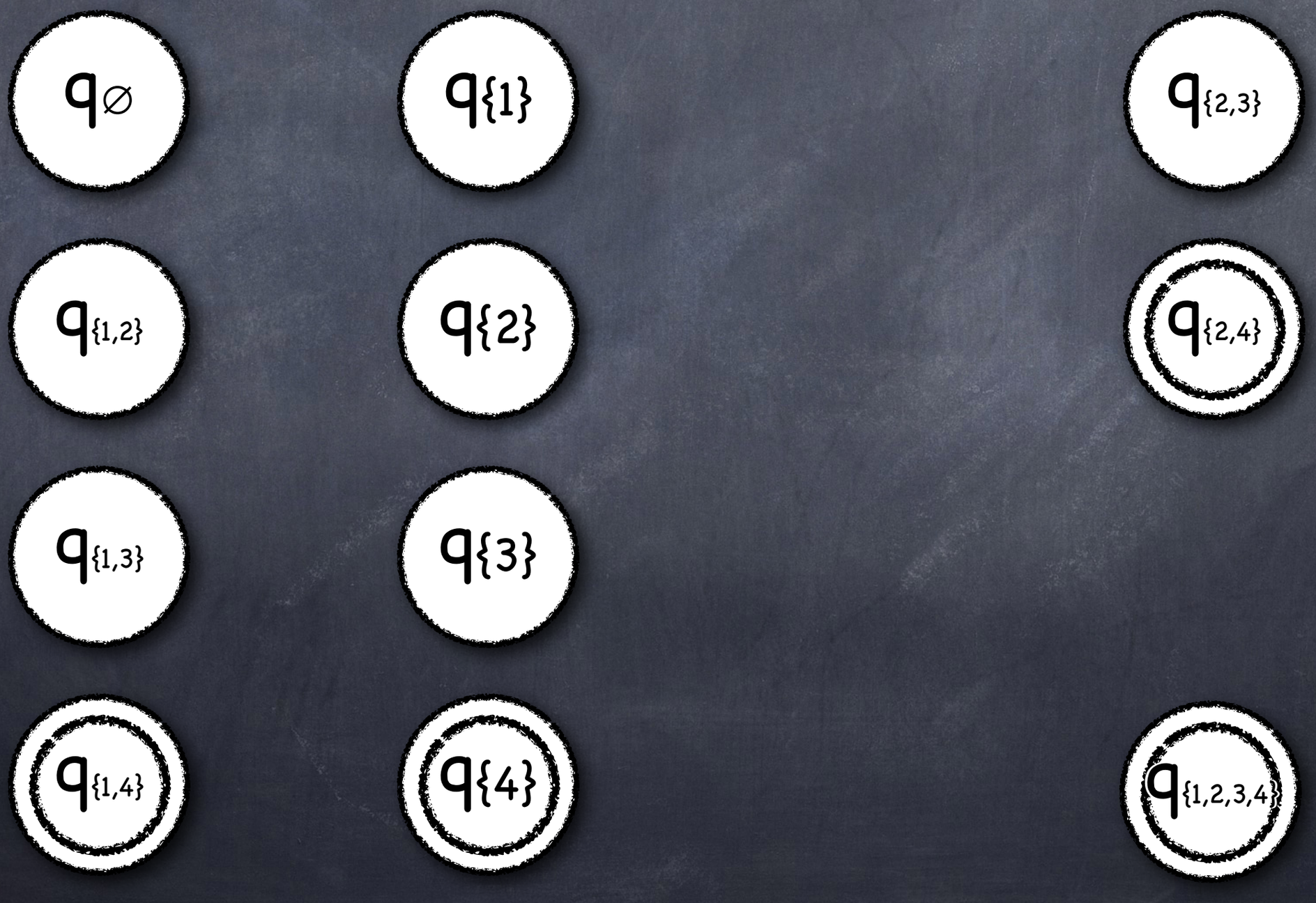
N_1



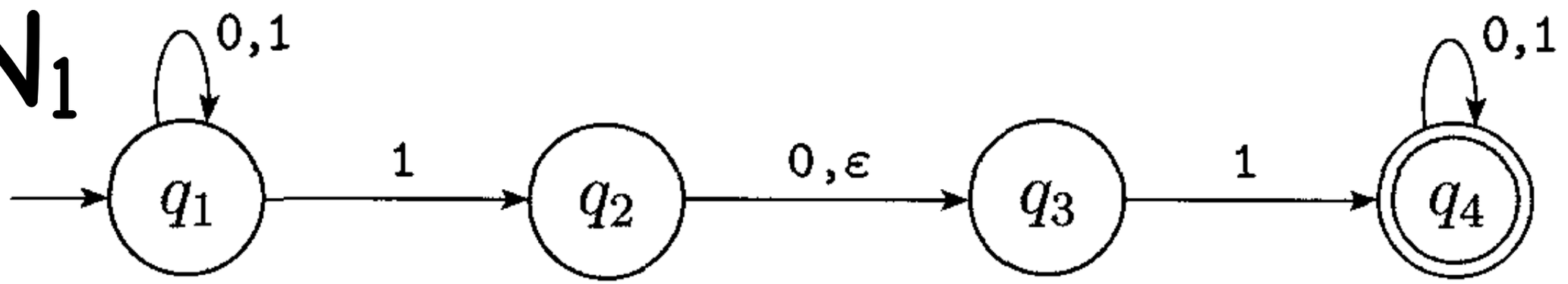


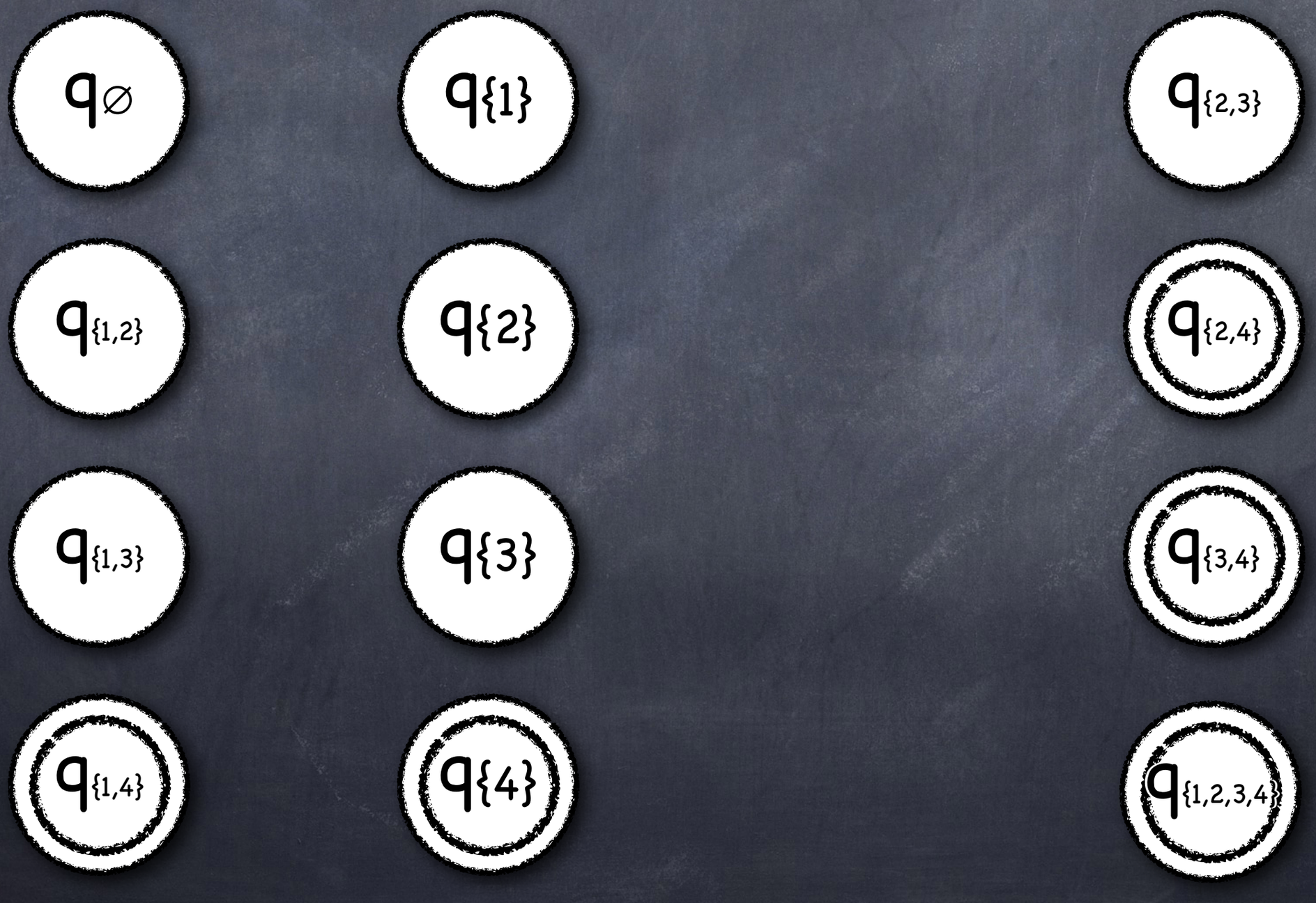
N_1



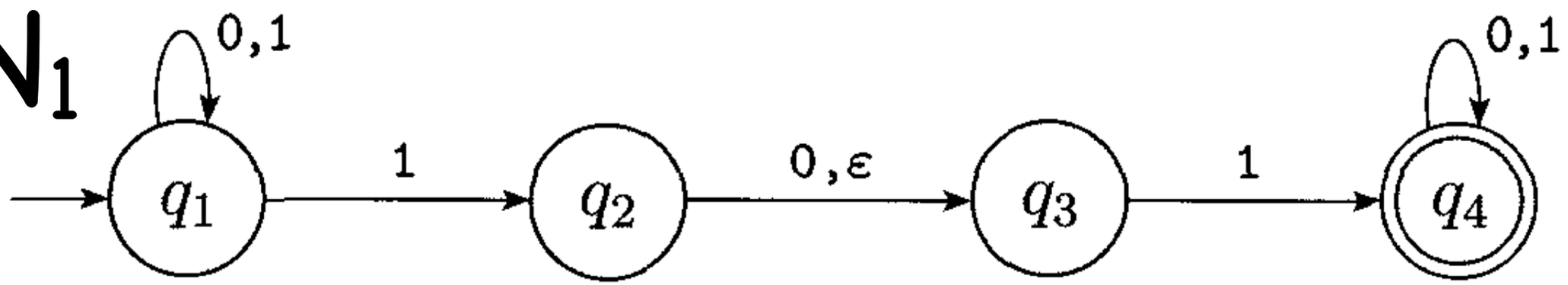


N_1





N_1



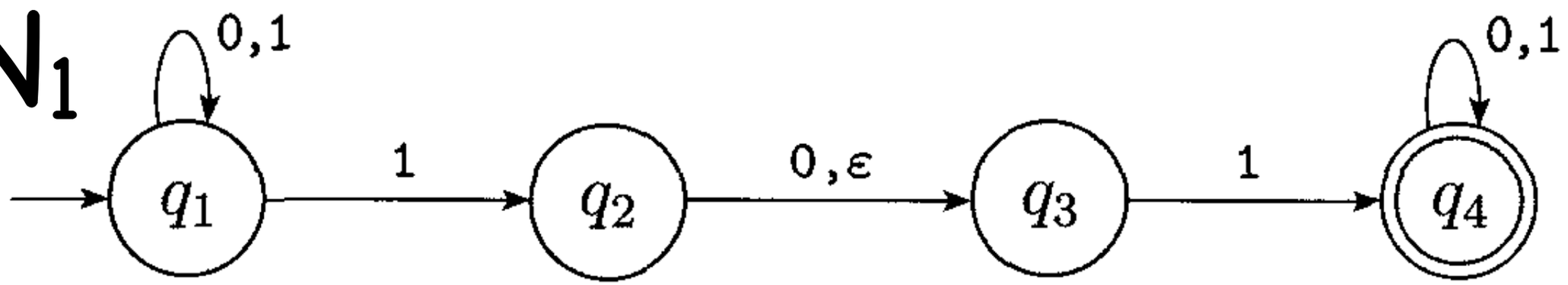
$q_\varnothing$ $q_{\{1\}}$ $q_{\{1,2,3\}}$ $q_{\{2,3\}}$

$q_{\{1,2\}}$ $q_{\{2\}}$ $q_{\{2,4\}}$

$q_{\{1,3\}}$ $q_{\{3\}}$ $q_{\{3,4\}}$

$q_{\{1,4\}}$ $q_{\{4\}}$ $q_{\{1,2,3,4\}}$

N_1



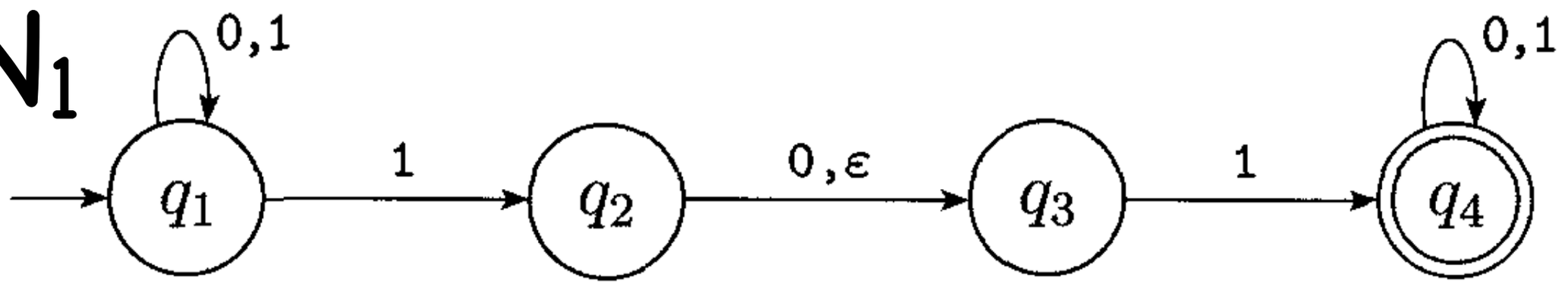
$q_\varnothing$ $q_{\{1\}}$ $q_{\{1,2,3\}}$ $q_{\{2,3\}}$

$q_{\{1,2\}}$ $q_{\{2\}}$ $q_{\{1,2,4\}}$ $q_{\{2,4\}}$

$q_{\{1,3\}}$ $q_{\{3\}}$ $q_{\{3,4\}}$

$q_{\{1,4\}}$ $q_{\{4\}}$ $q_{\{1,2,3,4\}}$

N_1



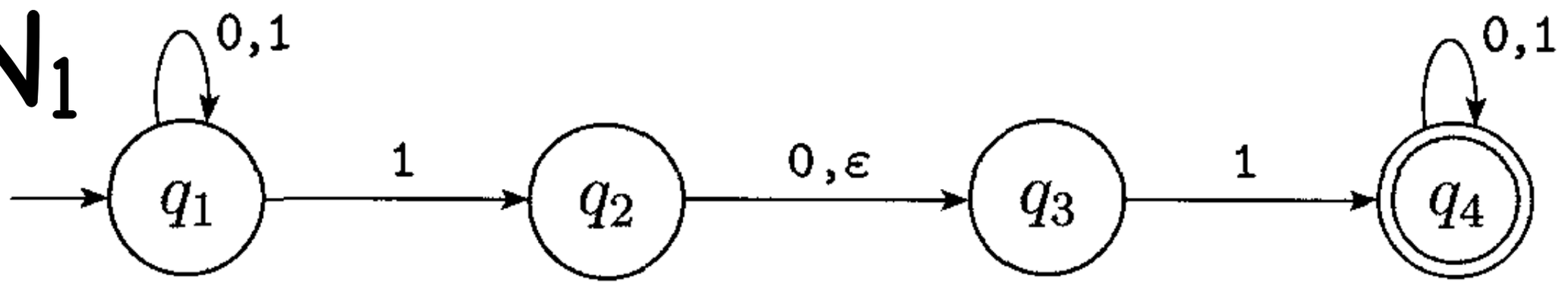
$q_\varnothing$ $q_{\{1\}}$ $q_{\{1,2,3\}}$ $q_{\{2,3\}}$

$q_{\{1,2\}}$ $q_{\{2\}}$ $q_{\{1,2,4\}}$ $q_{\{2,4\}}$

$q_{\{1,3\}}$ $q_{\{3\}}$ $q_{\{1,3,4\}}$ $q_{\{3,4\}}$

$q_{\{1,4\}}$ $q_{\{4\}}$ $q_{\{1,2,3,4\}}$

N_1



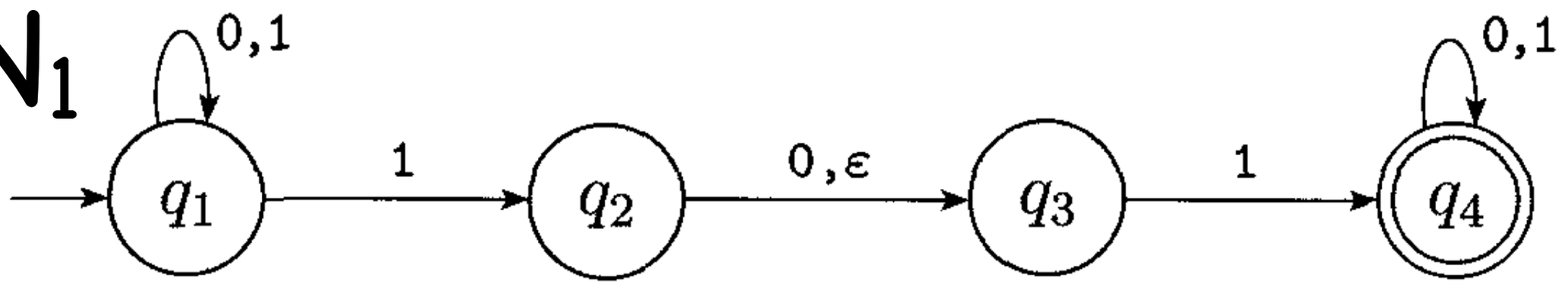
$q_\emptyset$ $q_{\{1\}}$ $q_{\{1,2,3\}}$ $q_{\{2,3\}}$

$q_{\{1,2\}}$ $q_{\{2\}}$ $q_{\{1,2,4\}}$ $q_{\{2,4\}}$

$q_{\{1,3\}}$ $q_{\{3\}}$ $q_{\{1,3,4\}}$ $q_{\{3,4\}}$

$q_{\{1,4\}}$ $q_{\{4\}}$ $q_{\{2,3,4\}}$ $q_{\{1,2,3,4\}}$

N_1

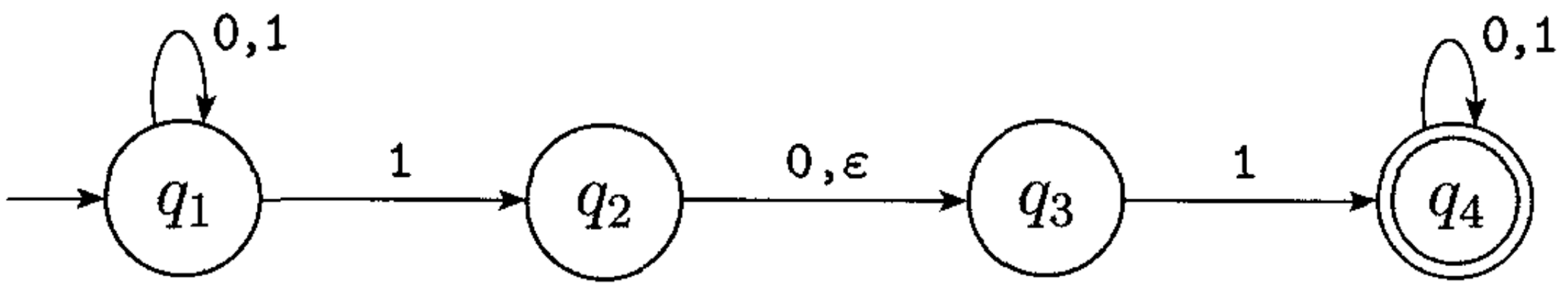


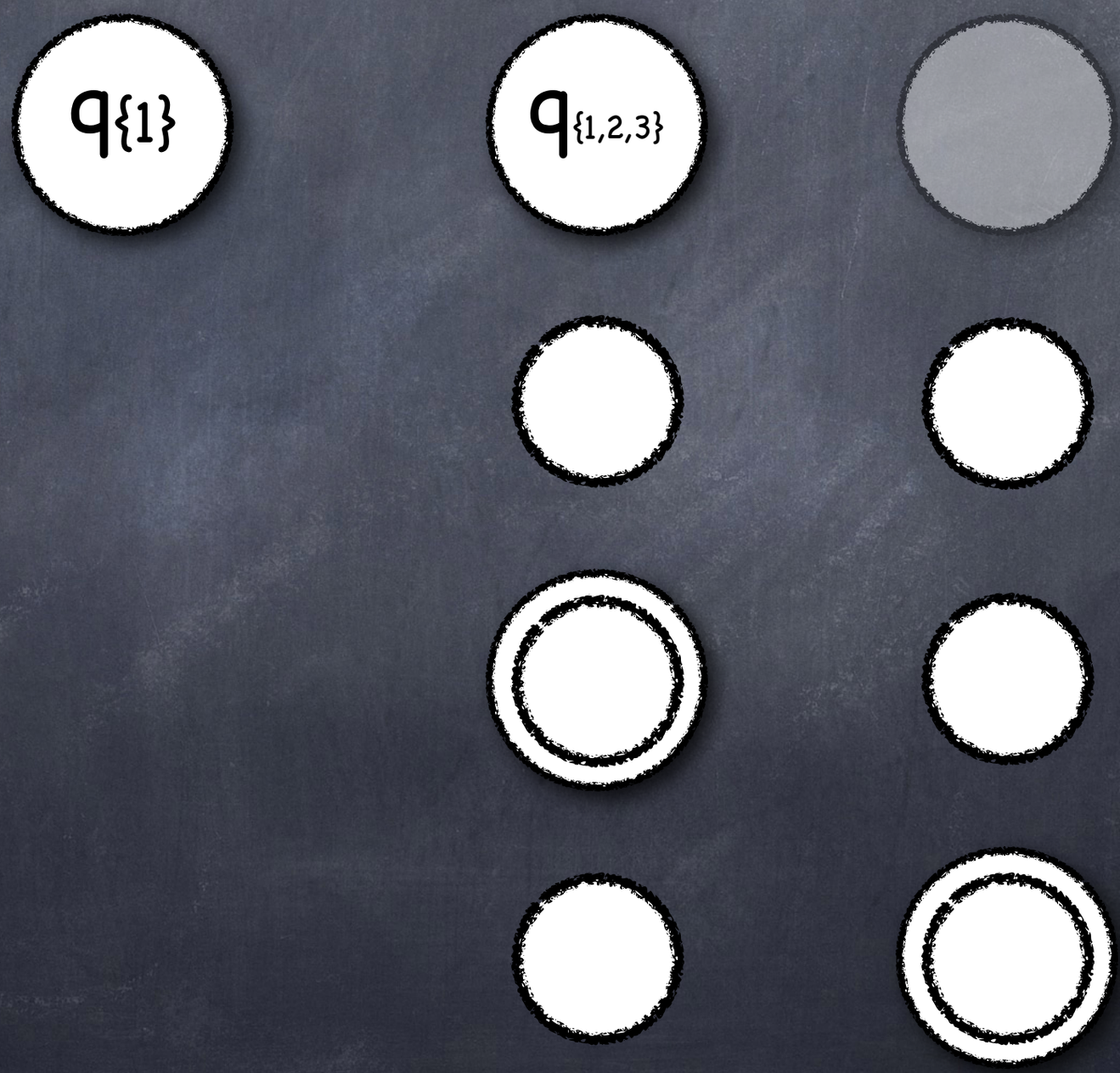
$q_\varnothing$ $q_{\{1\}}$ $q_{\{1,2,3\}}$ $q_{\{2,3\}}$

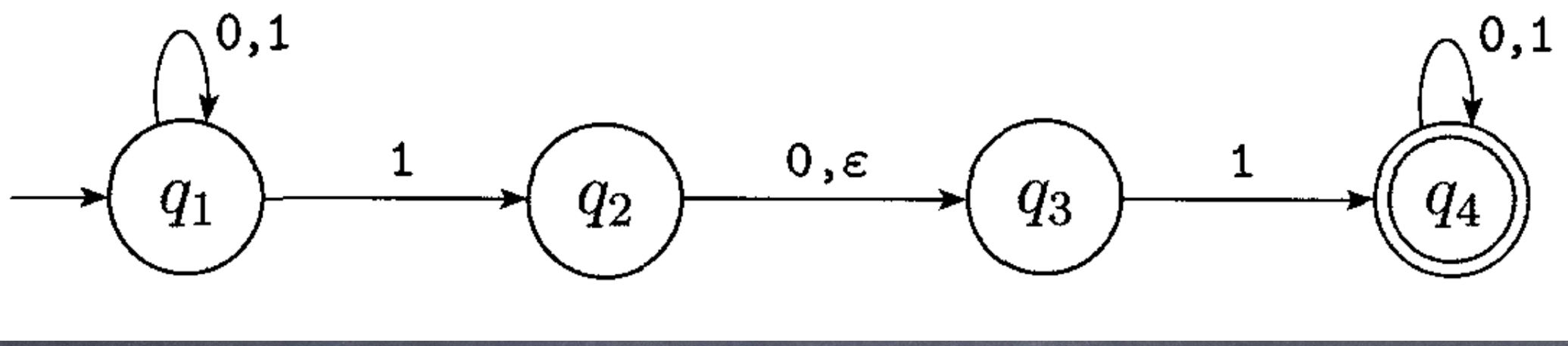
$q_{\{1,2\}}$ $q_{\{2\}}$ $q_{\{1,2,4\}}$ $q_{\{2,4\}}$

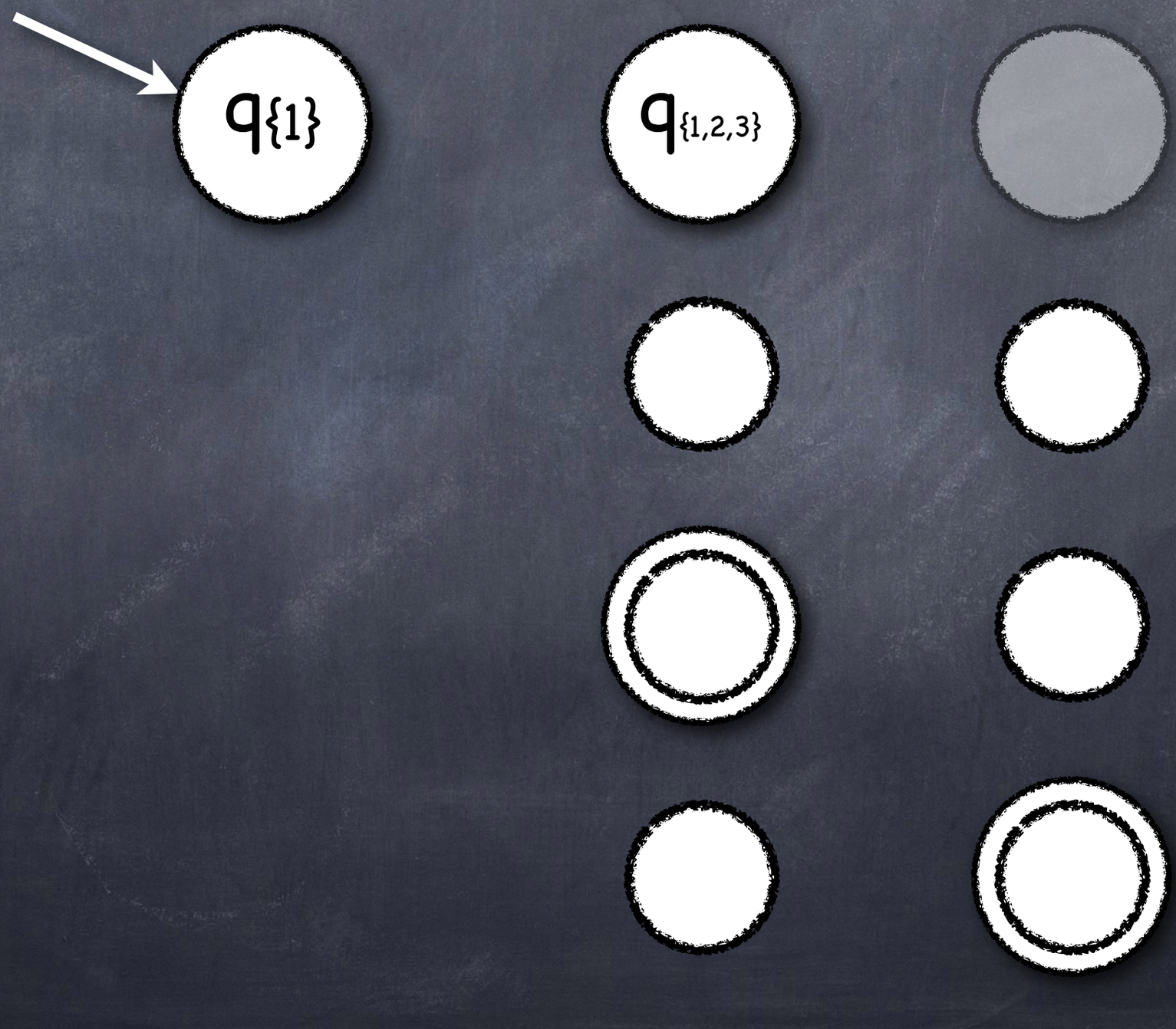
$q_{\{1,3\}}$ $q_{\{3\}}$ $q_{\{1,3,4\}}$ $q_{\{3,4\}}$

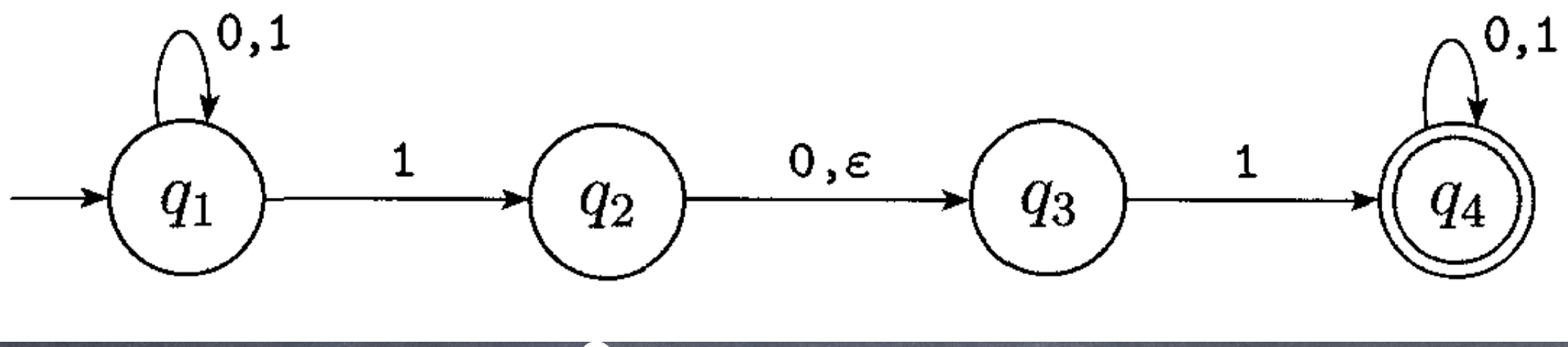
$q_{\{1,4\}}$ $q_{\{4\}}$ $q_{\{2,3,4\}}$ $q_{\{1,2,3,4\}}$

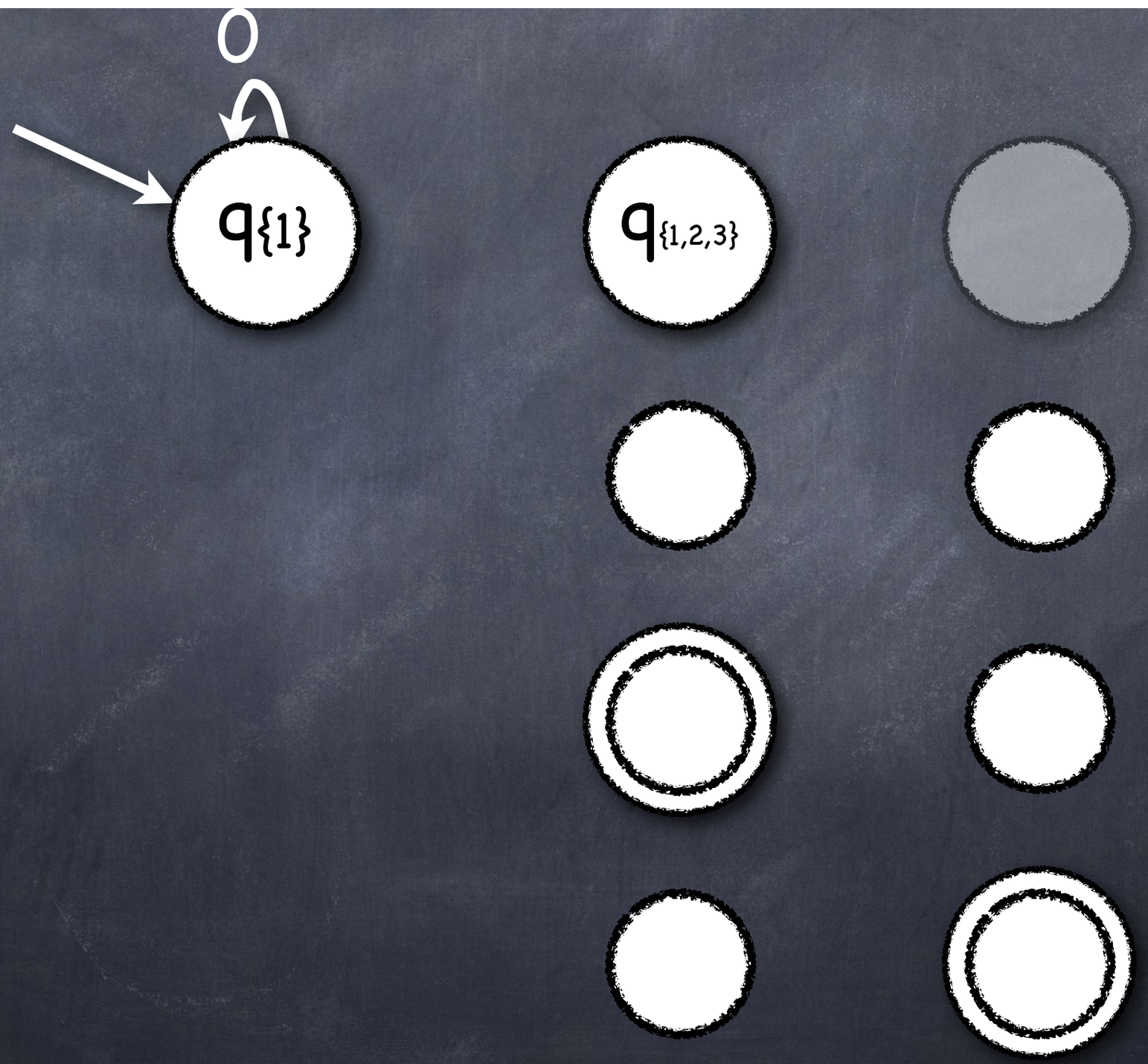

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

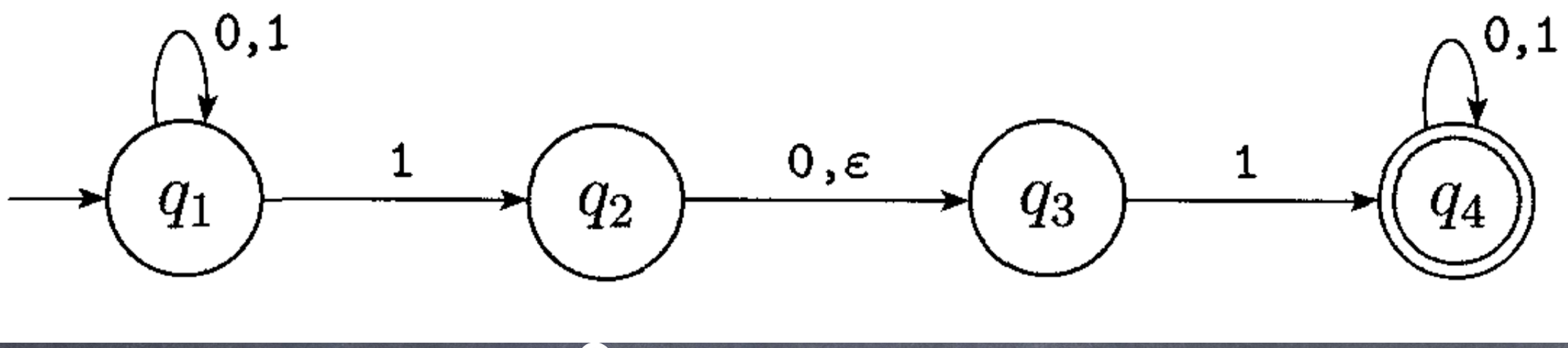

q{1}
q{1,2,3}

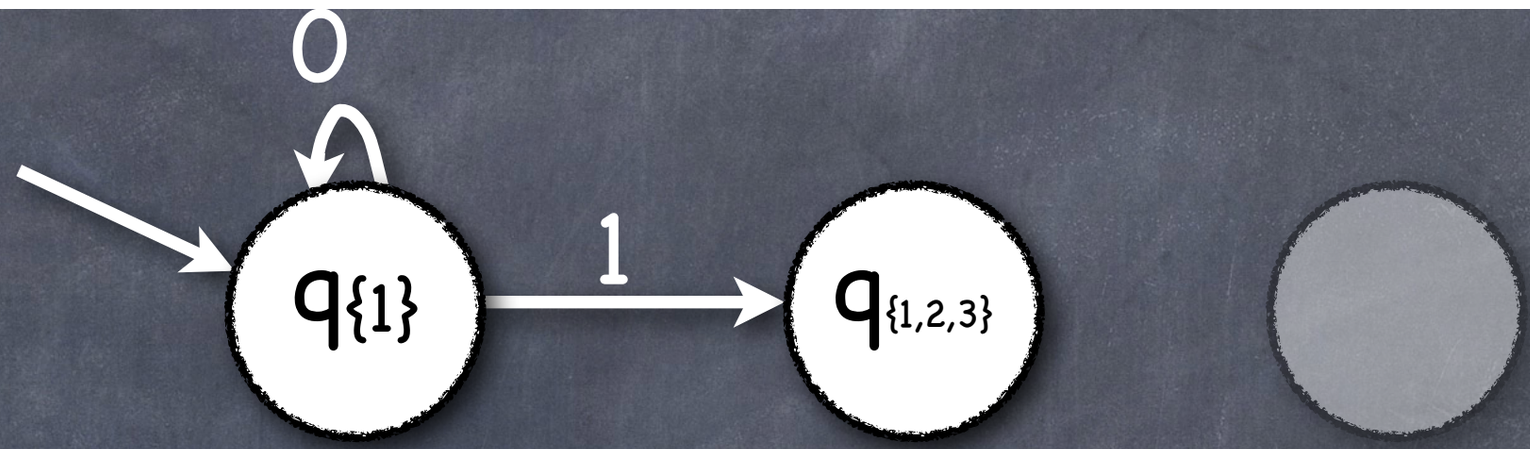

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

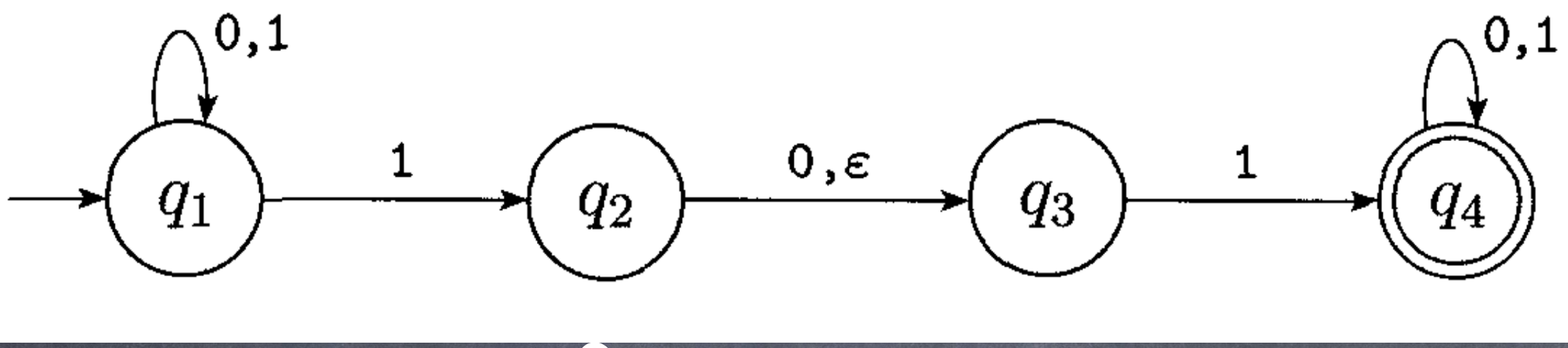

q{1}
q{1,2,3}

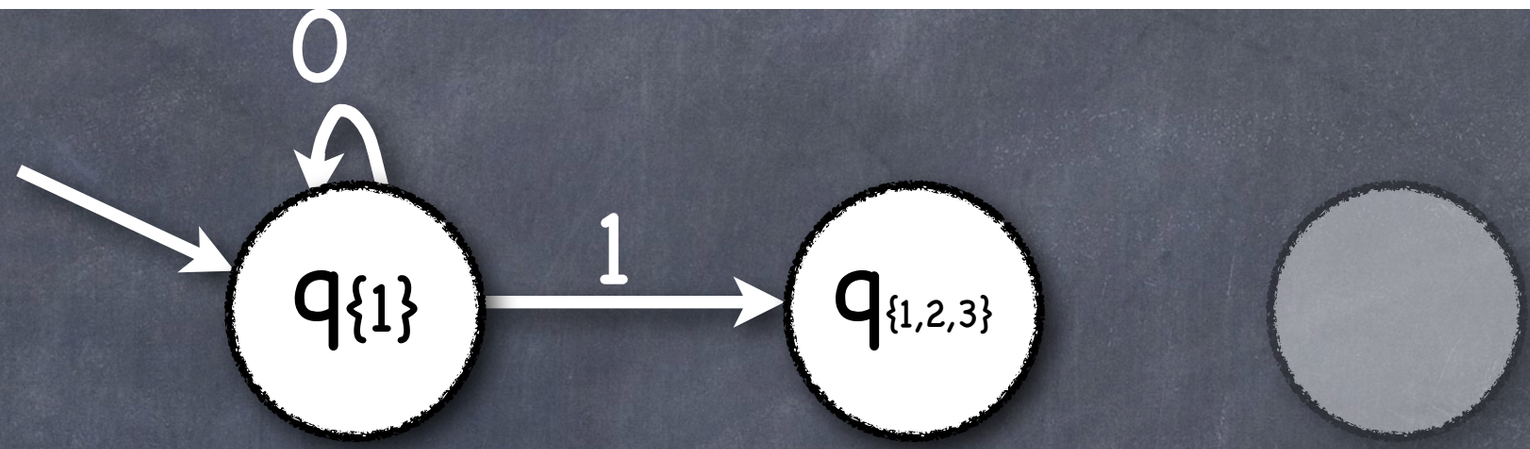

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

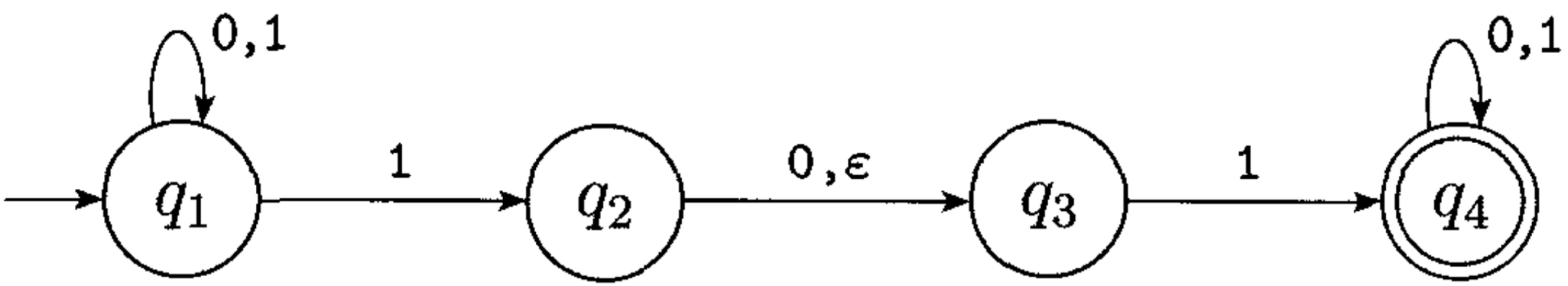

0
q{1}
q{1,2,3}

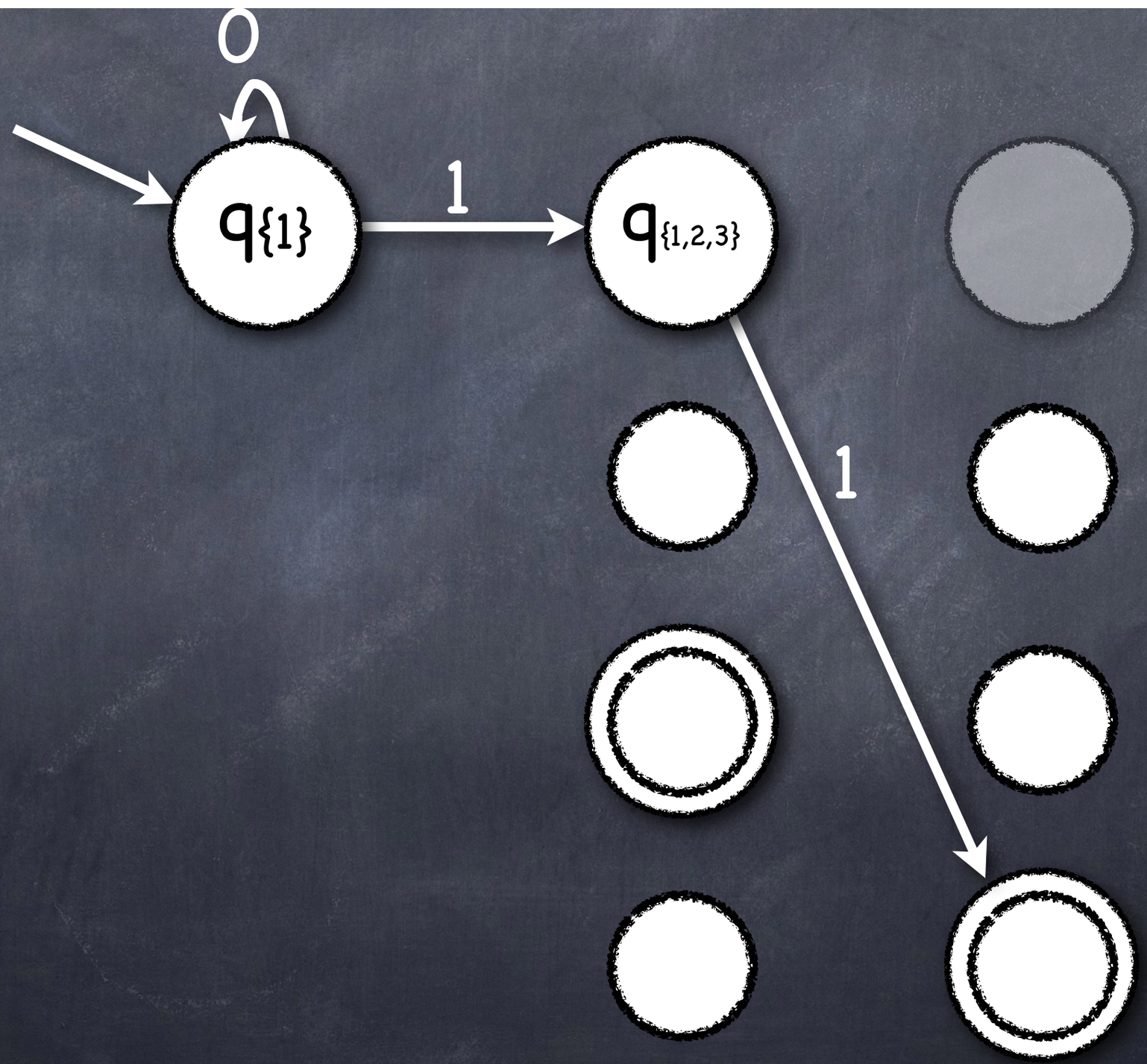

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

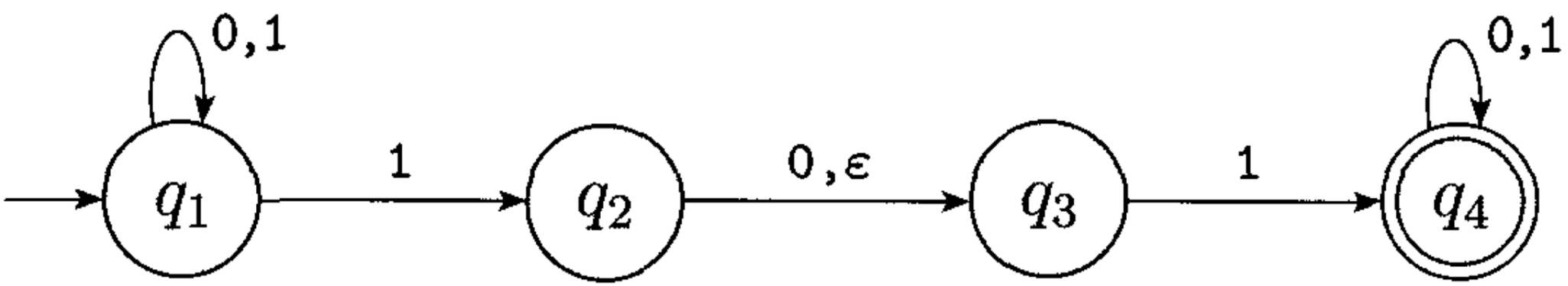

0
1
q{1}
q{1,2,3}

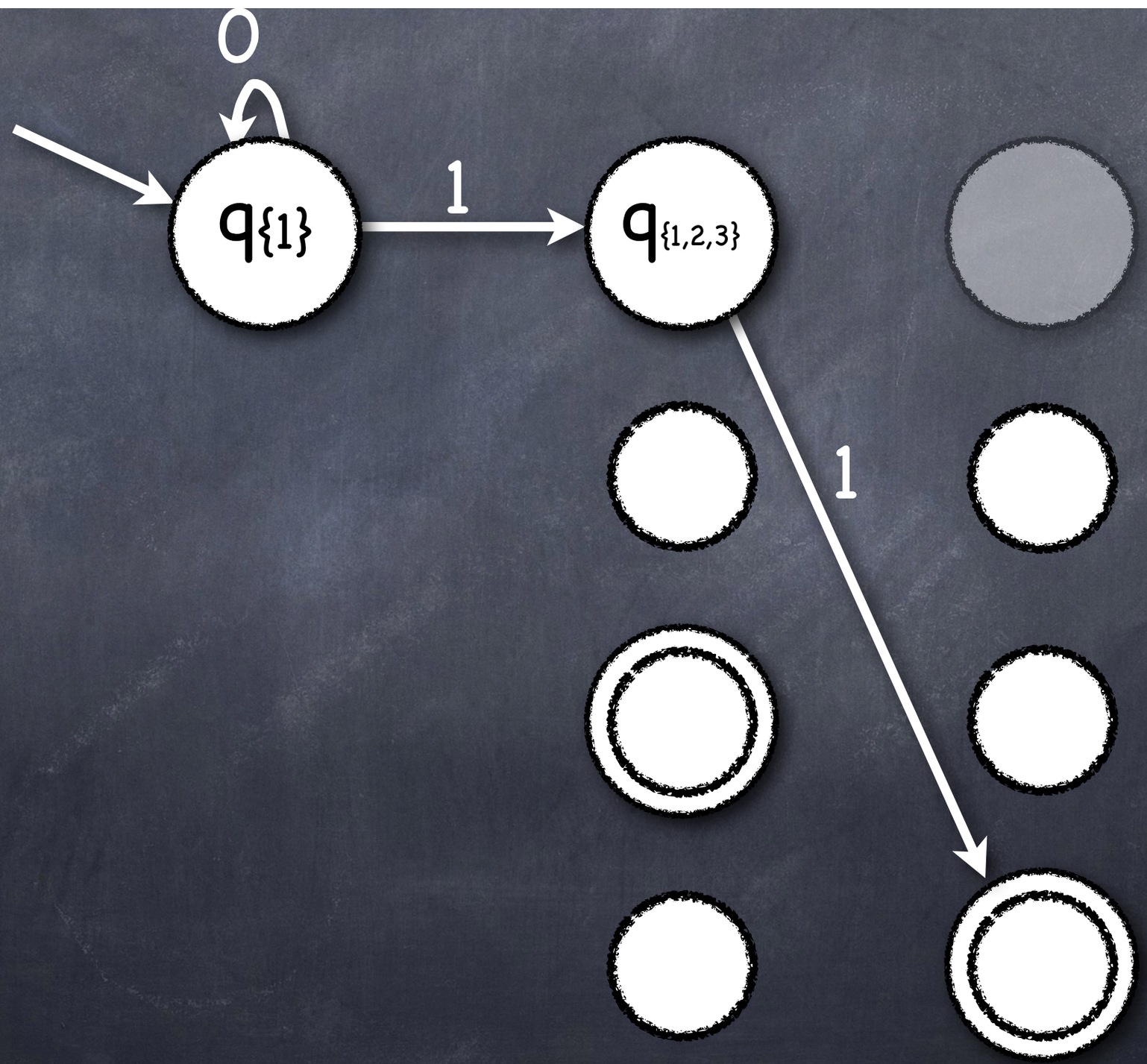

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

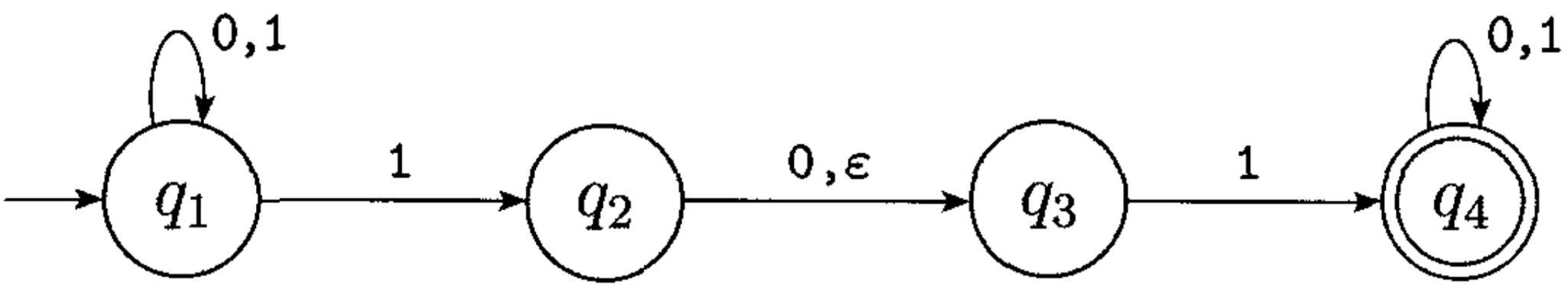

0
1
q{1}
q{1,2,3}

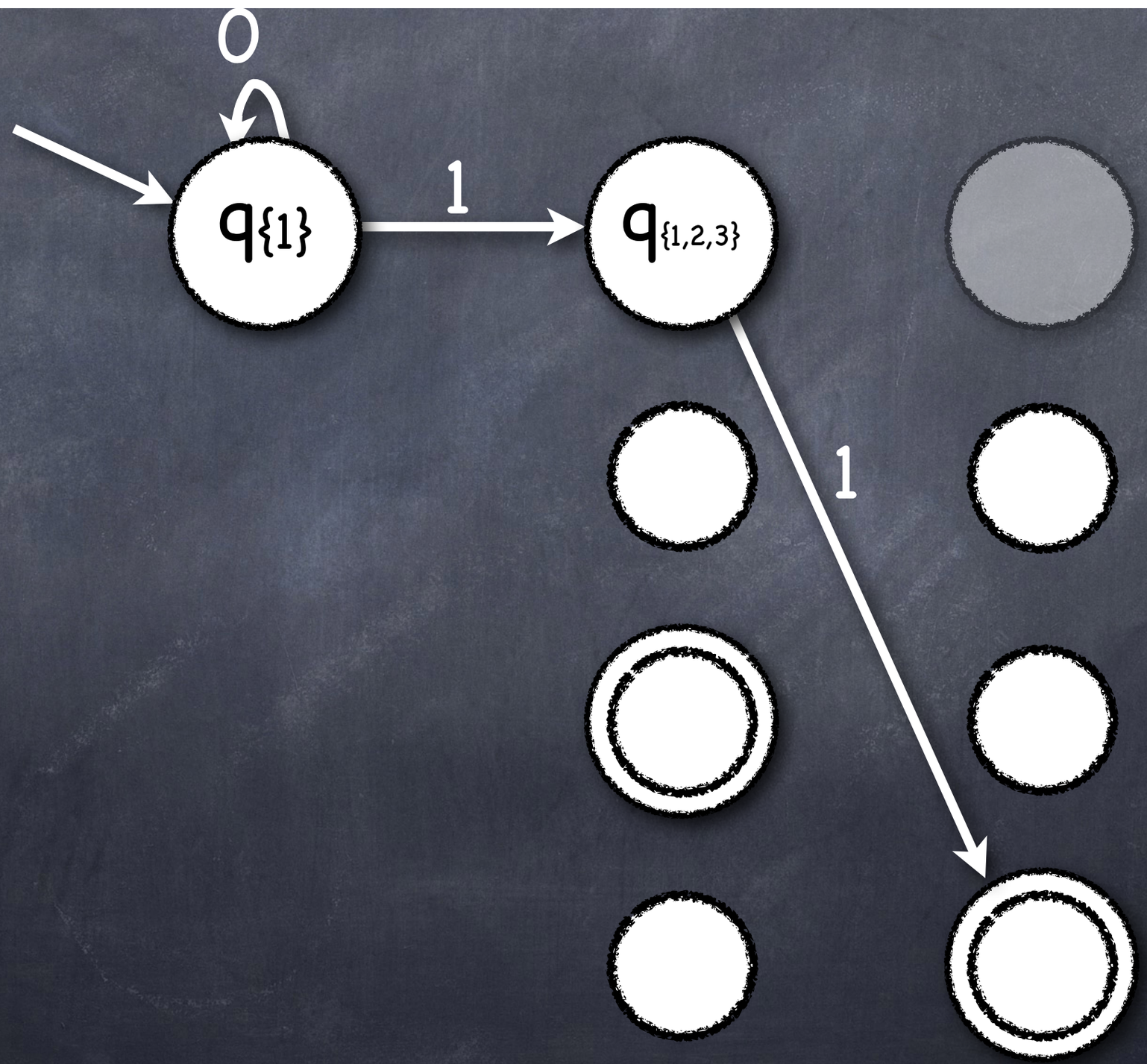

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

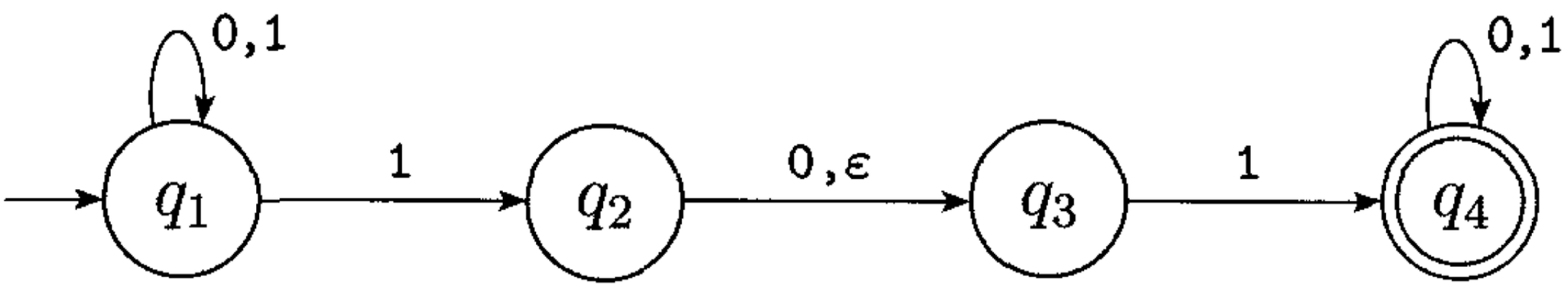

0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

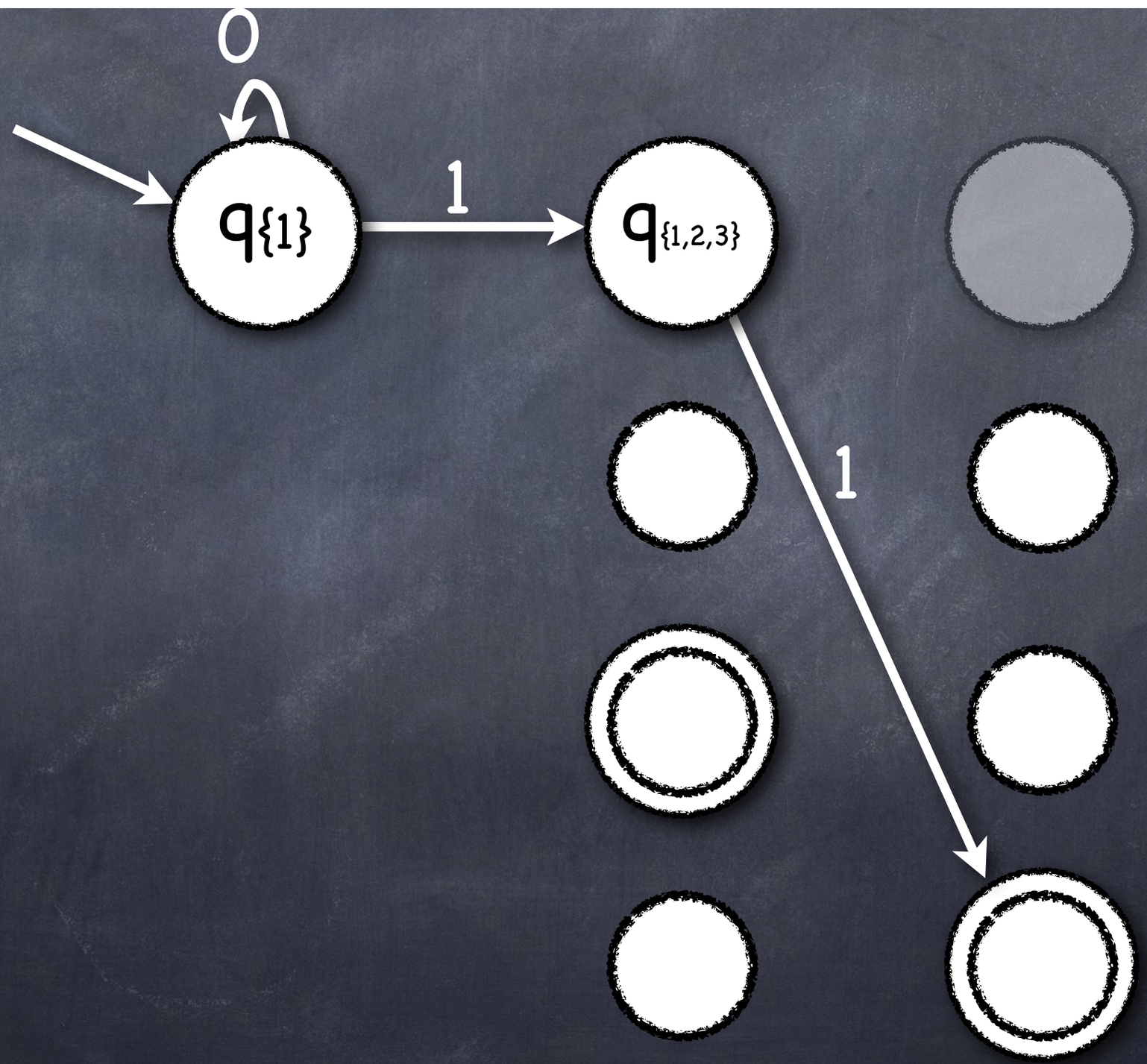

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

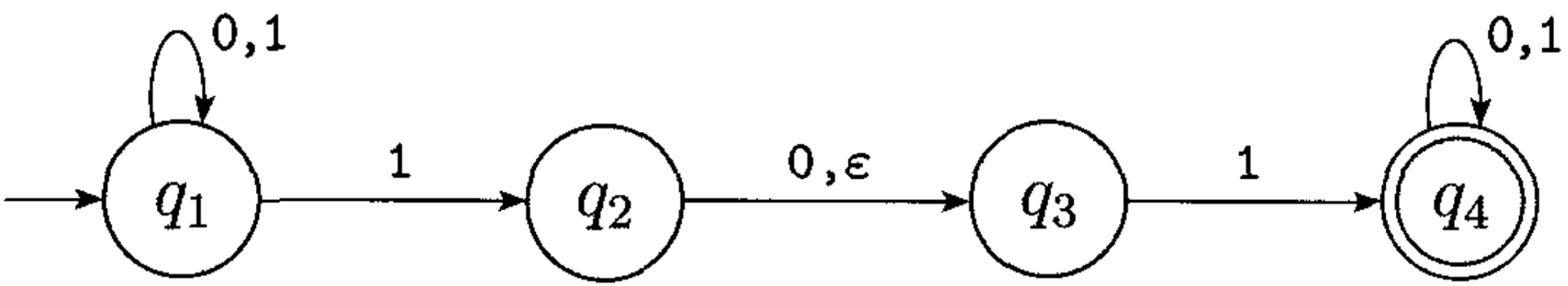

0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

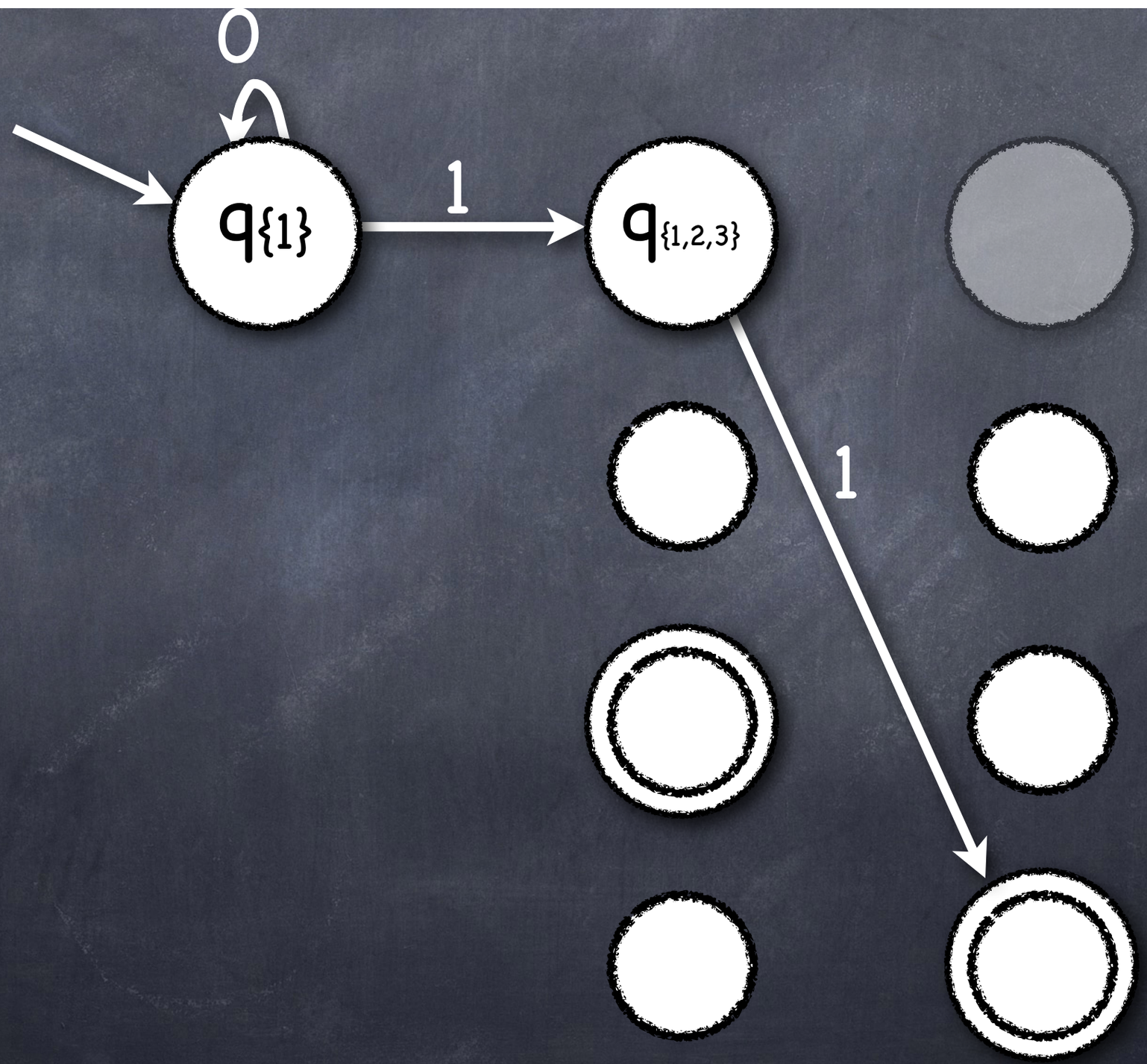

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

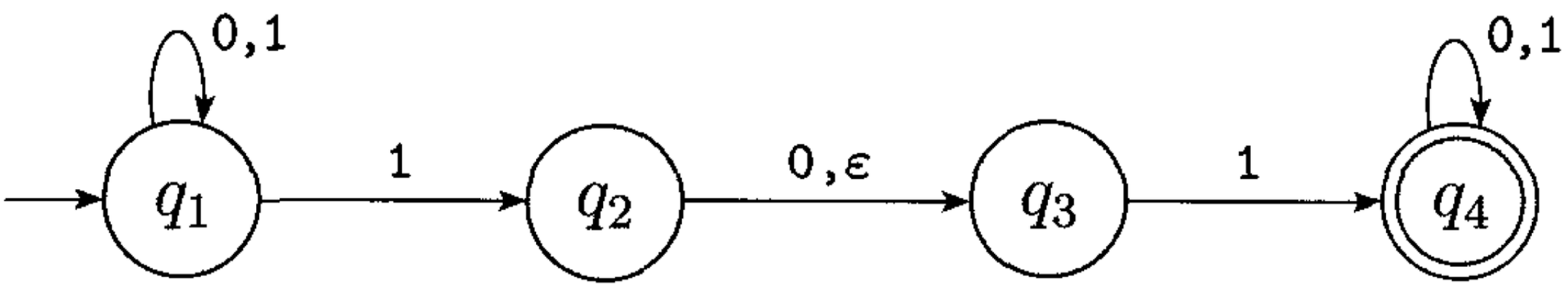

0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

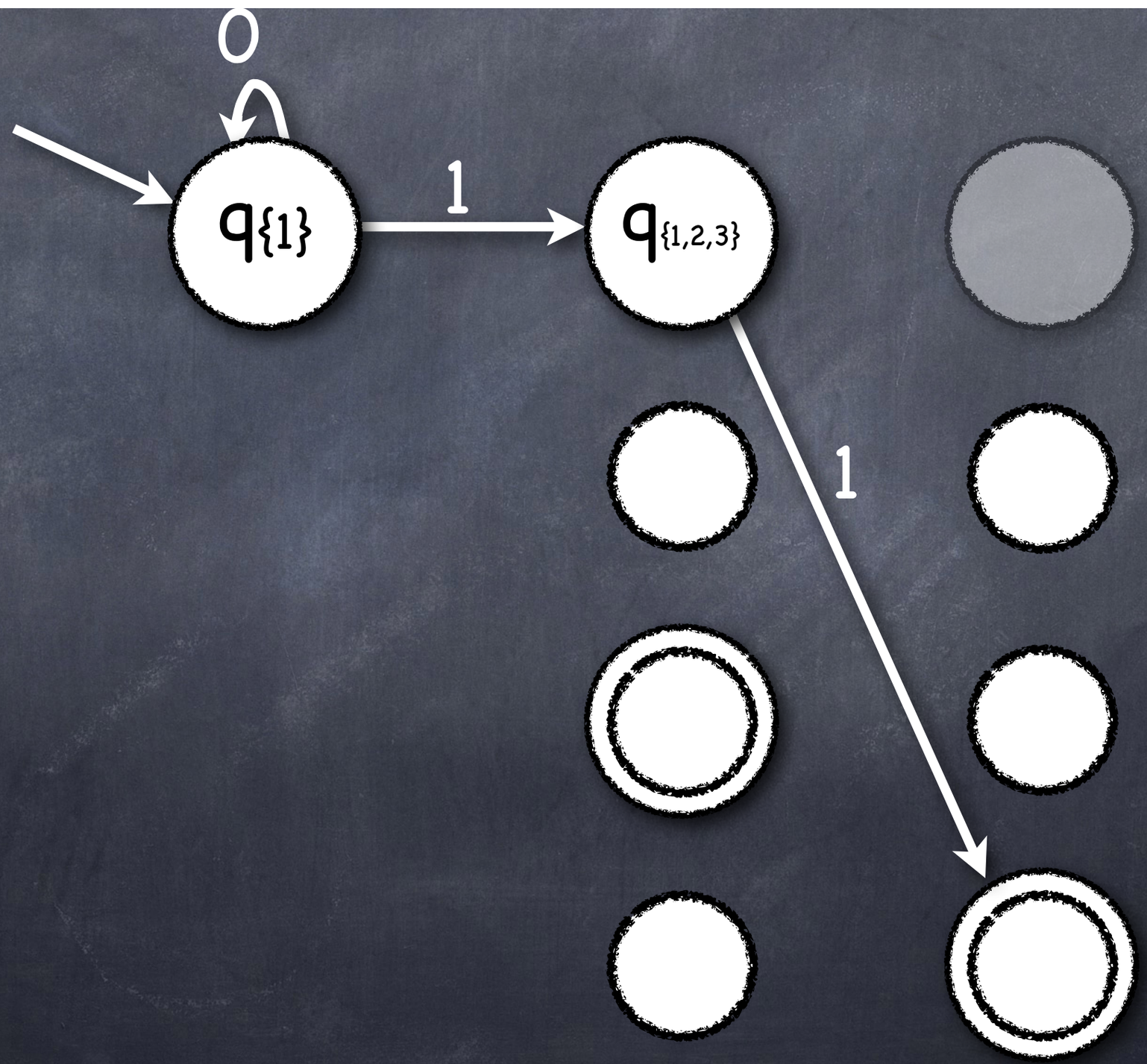

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

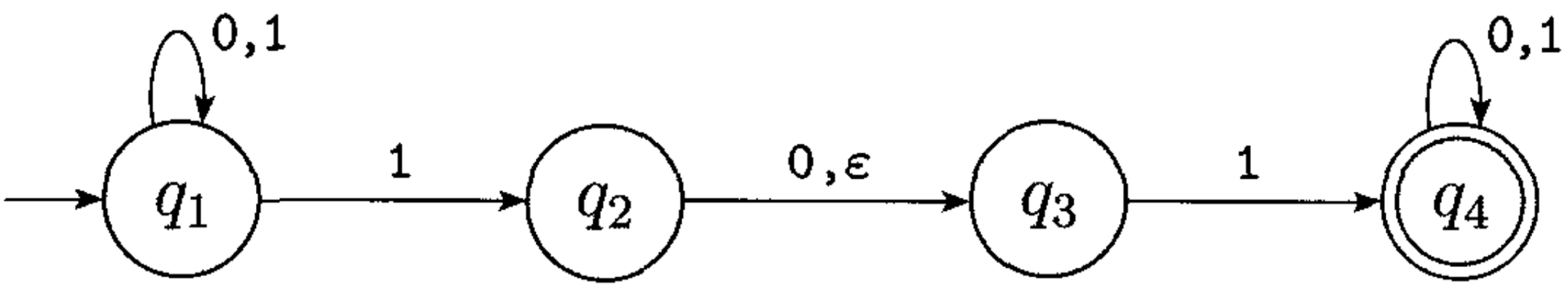

0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

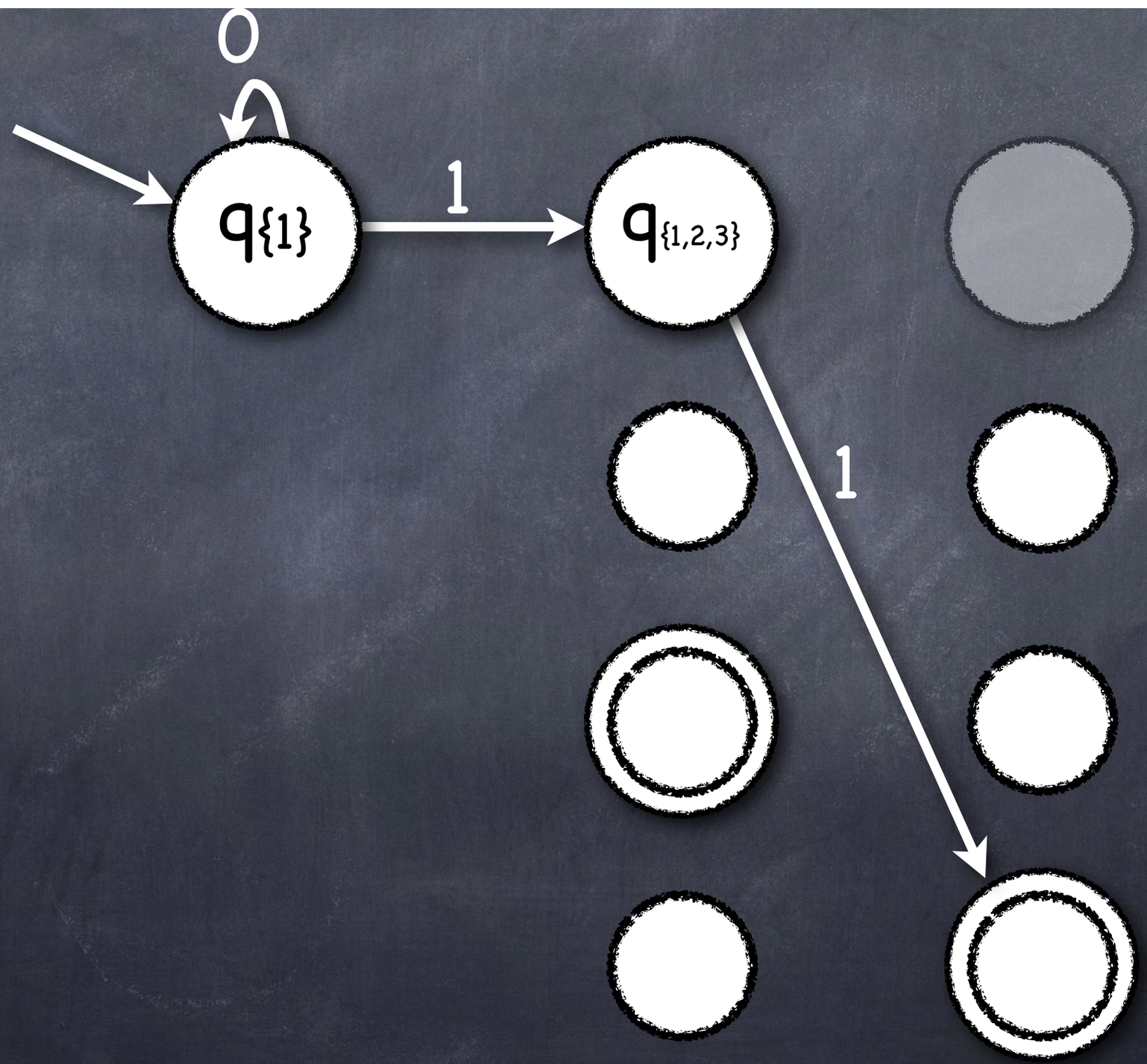

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

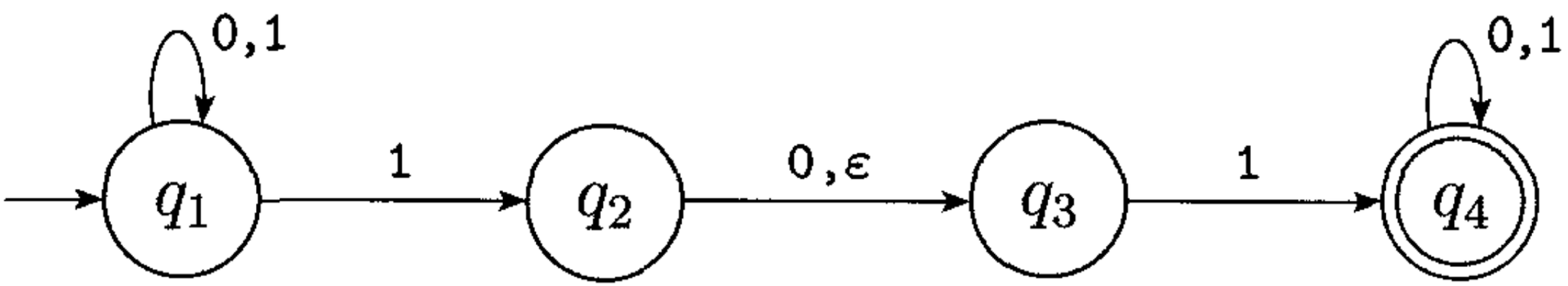

0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

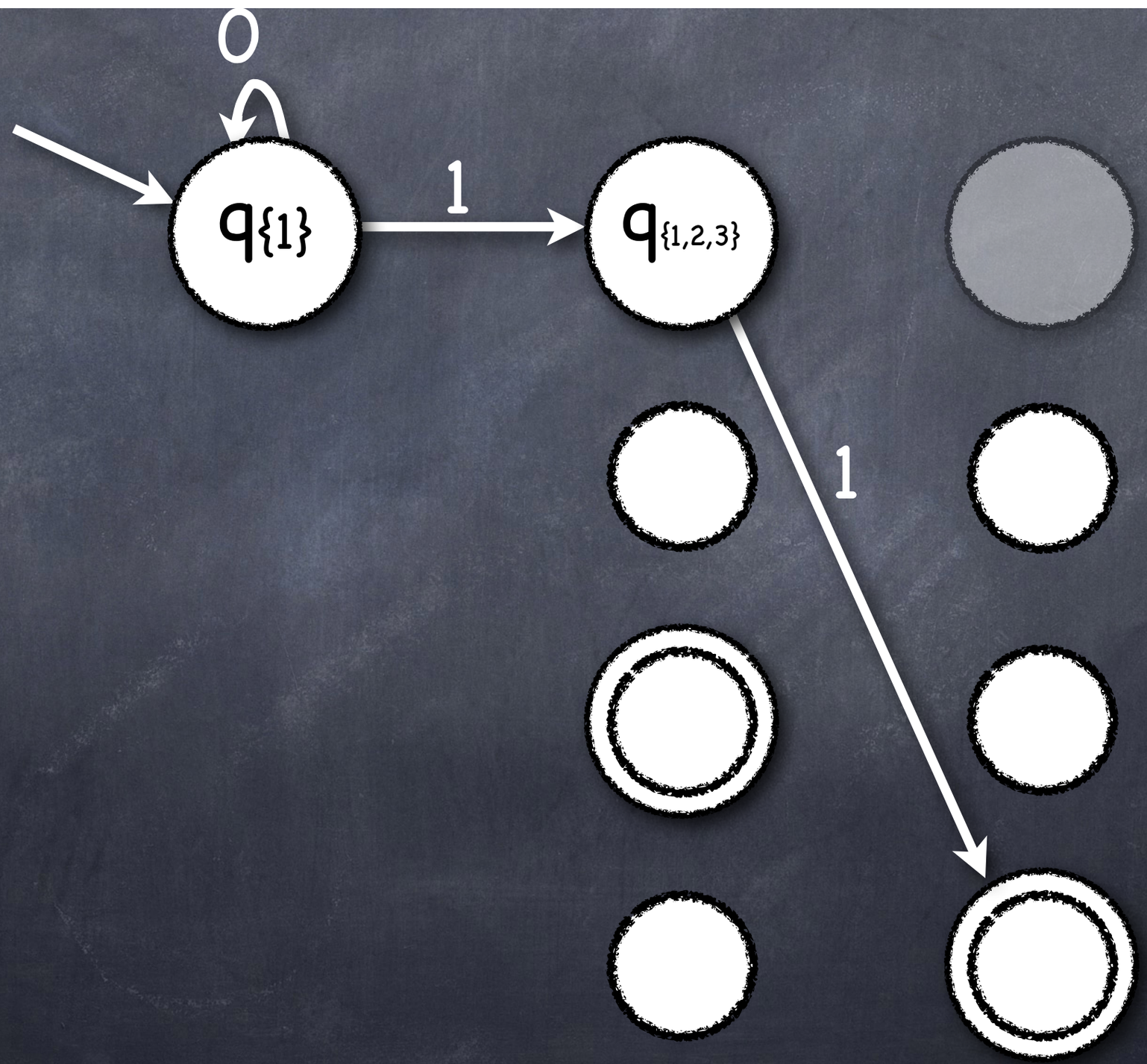

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

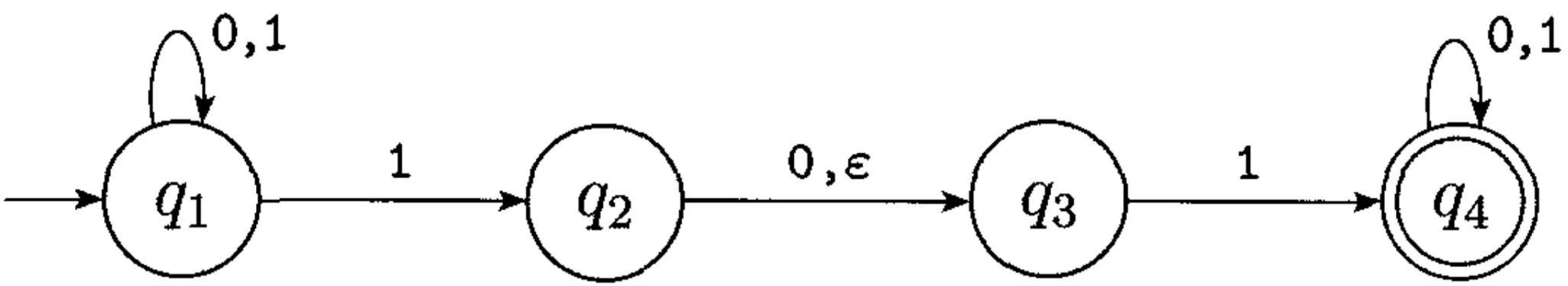

0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

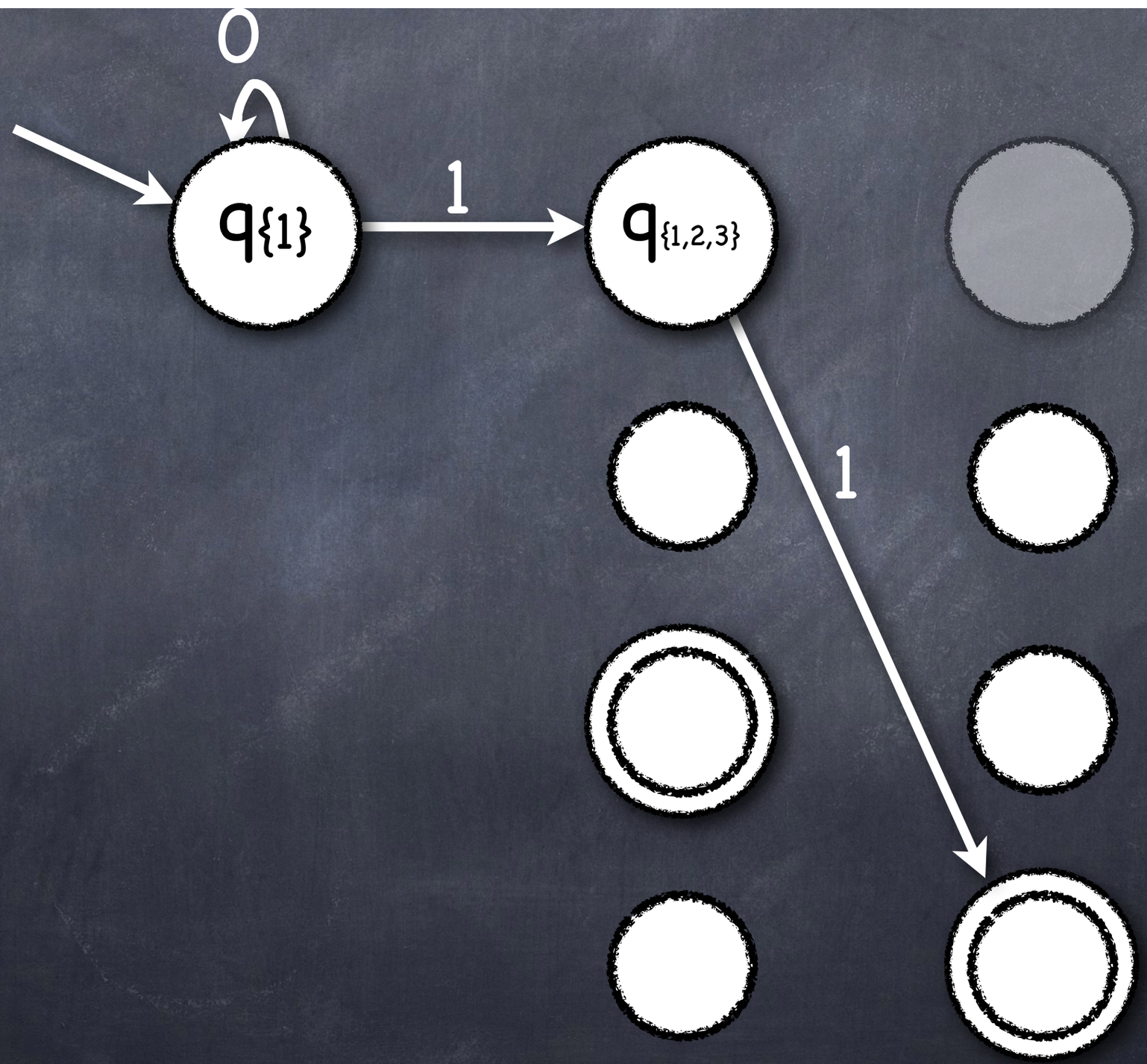

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4


0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

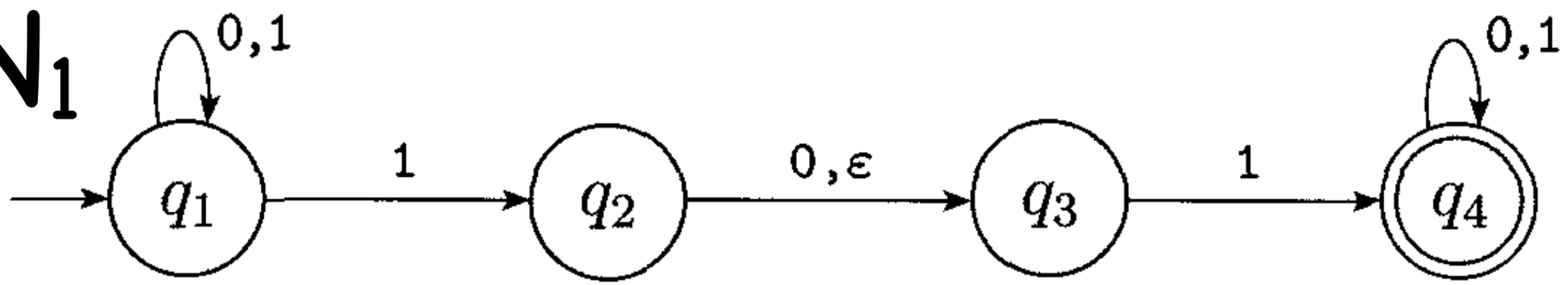

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4


0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

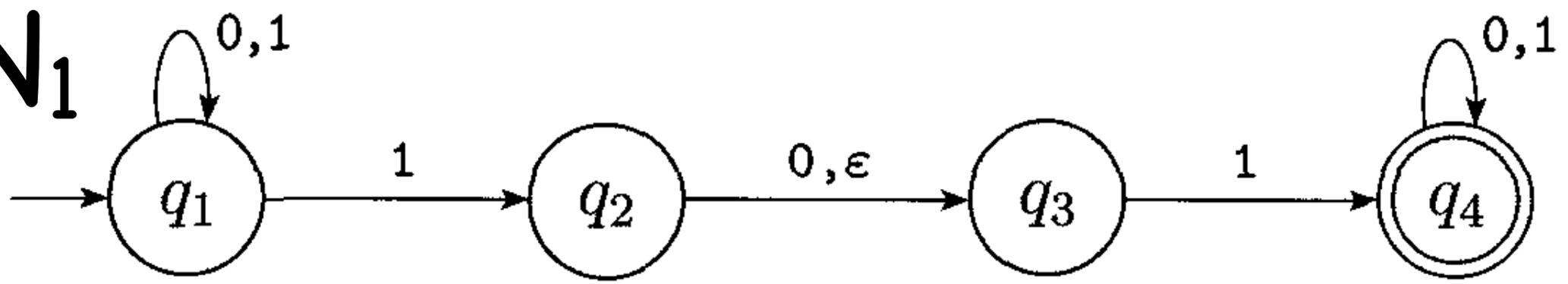

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4


0
1
1
$q_{\{1\}}$
$q_{\{1,2,3\}}$

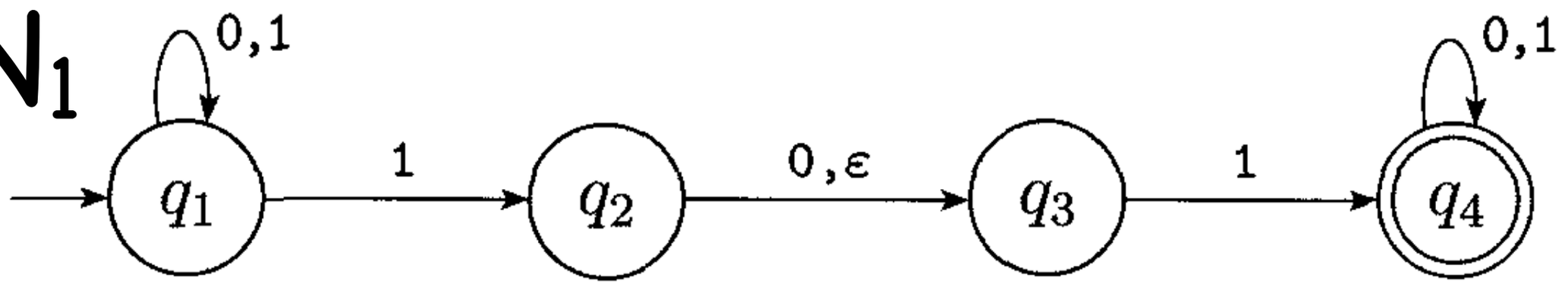
N_1



N_1



N_1



0

$q_{\{1\}}$

$q_{\{1,2,3\}}$

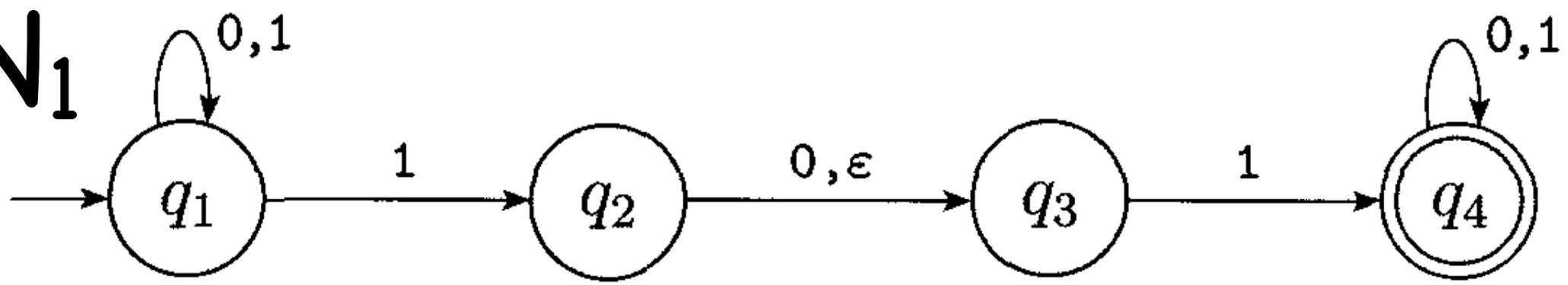
$q_{\{1,3\}}$

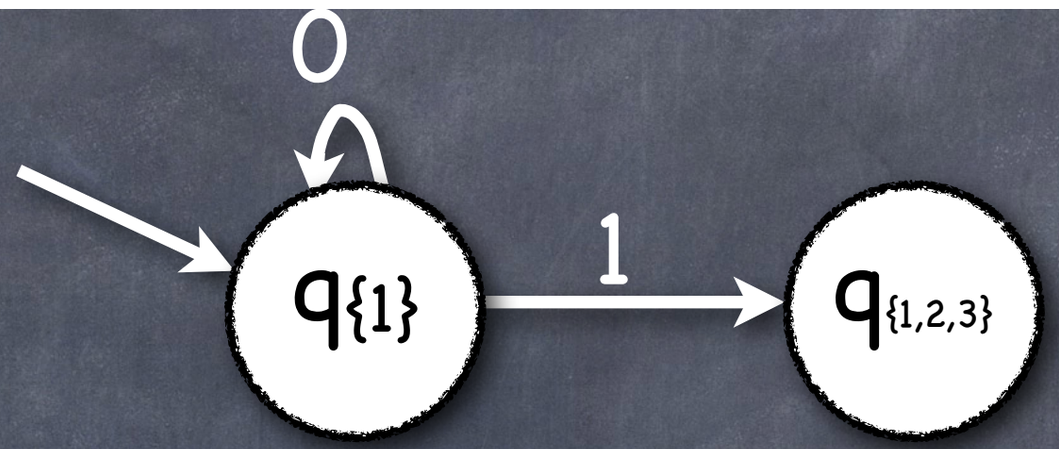
$q_{\{1,3,4\}}$

$q_{\{1,4\}}$

$q_{\{1,2,3,4\}}$

N_1





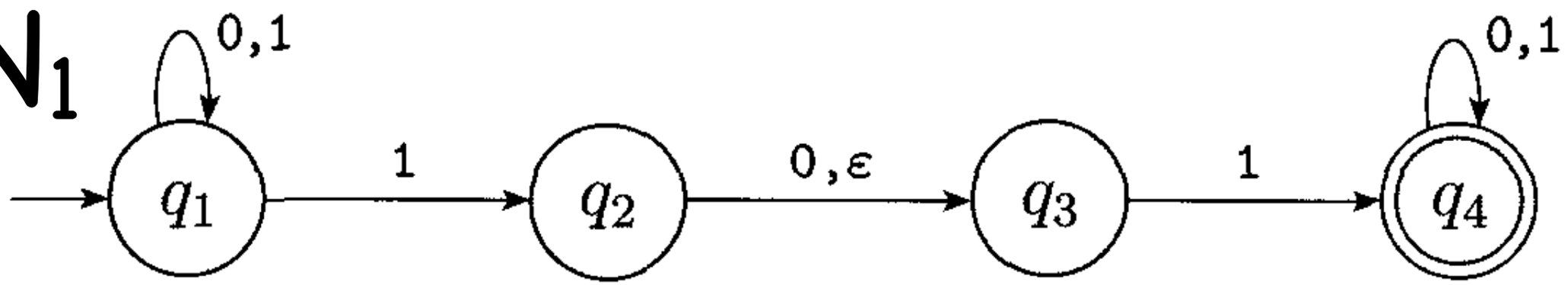
$q_{\{1,3\}}$

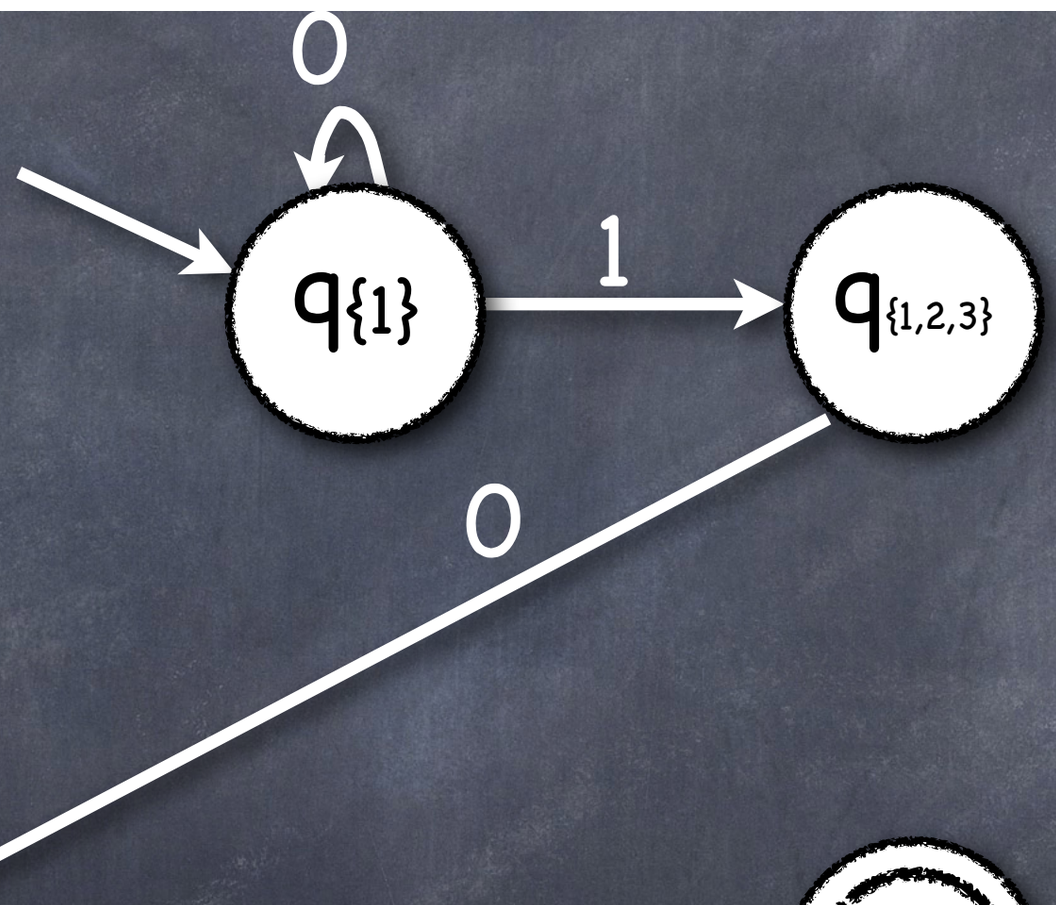
$q_{\{1,3,4\}}$

$q_{\{1,4\}}$

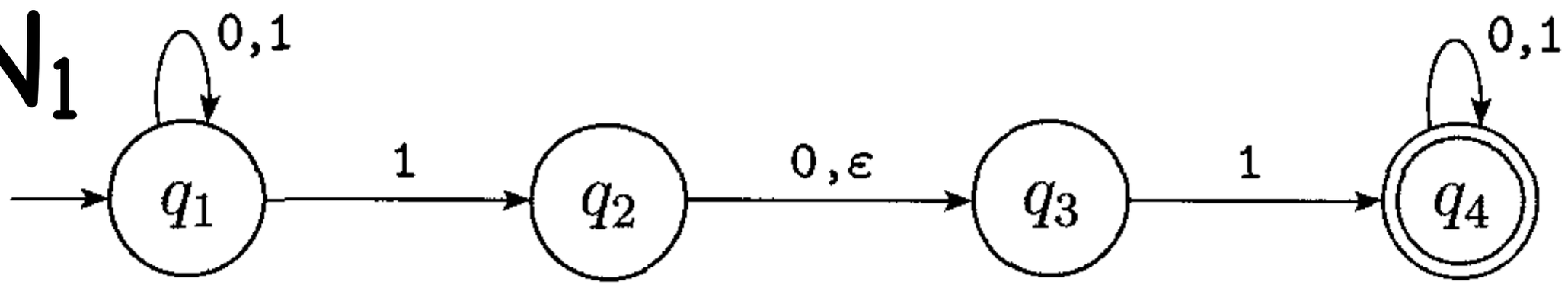
$q_{\{1,2,3,4\}}$

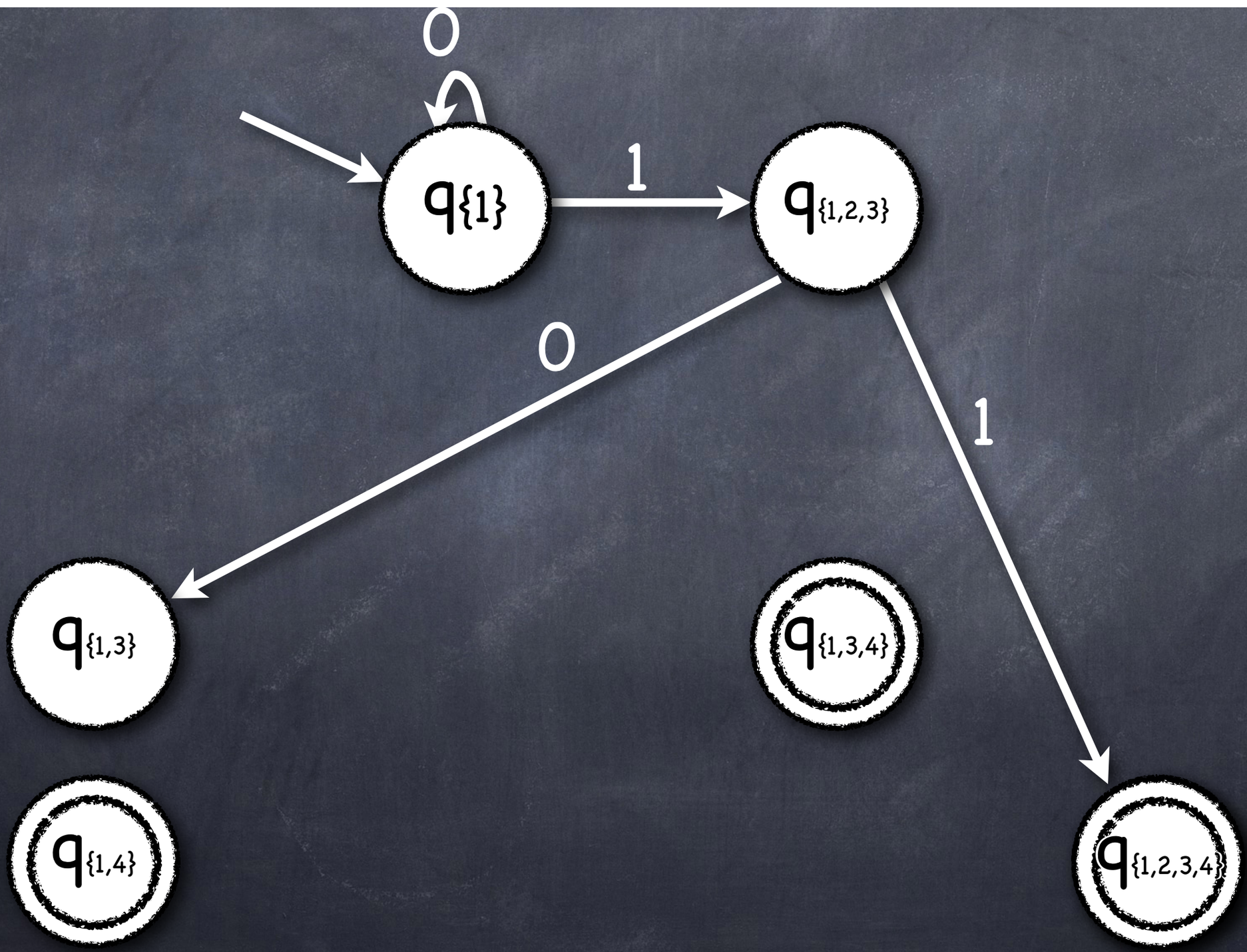
N_1



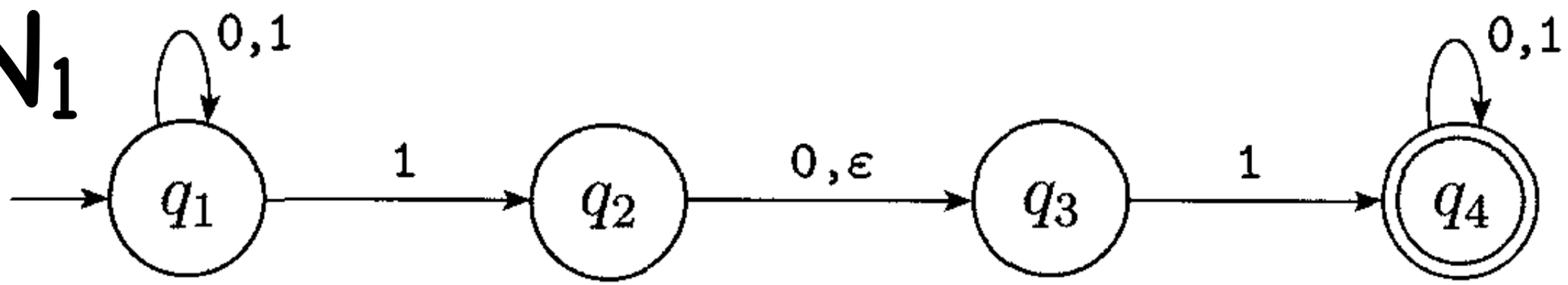


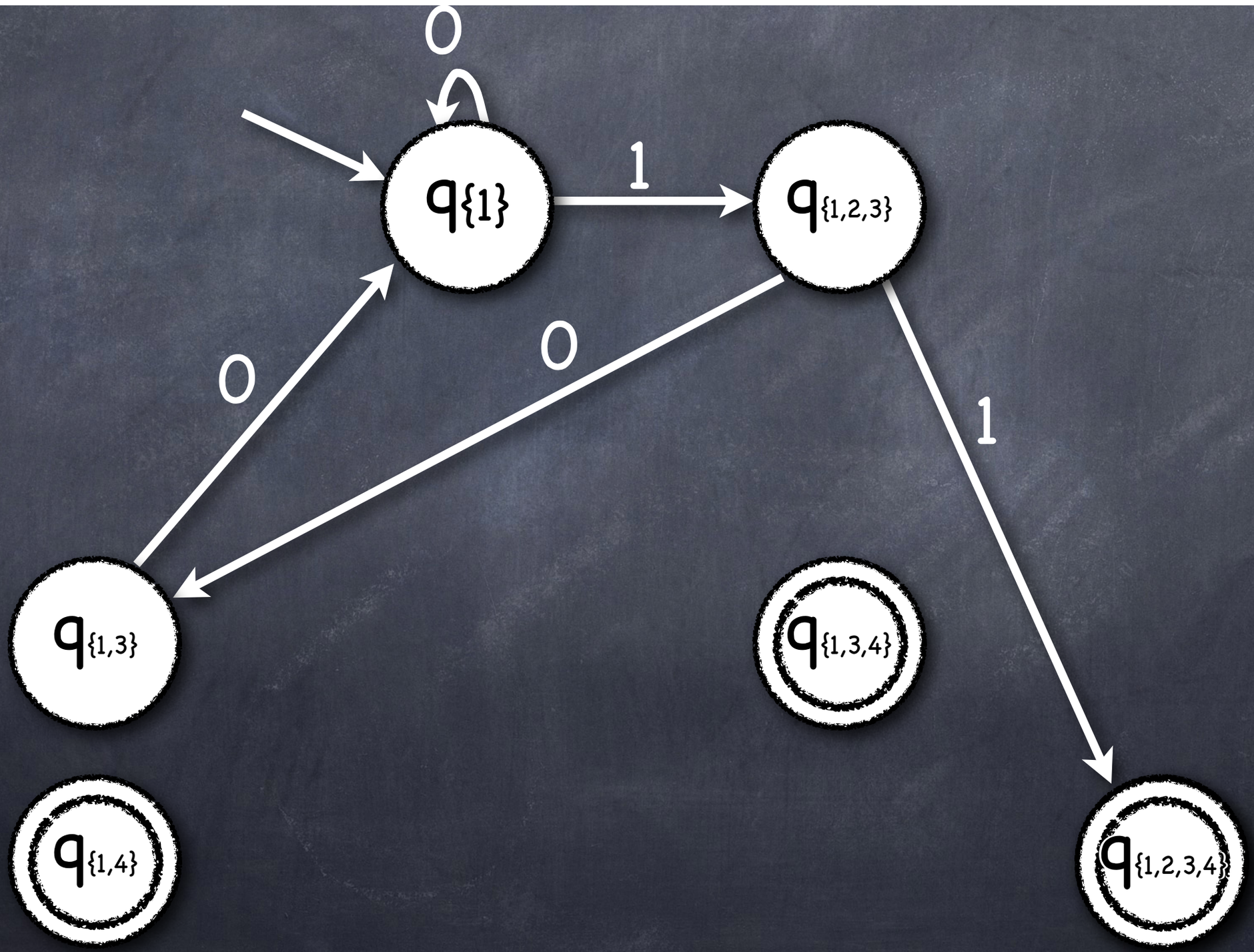
N_1



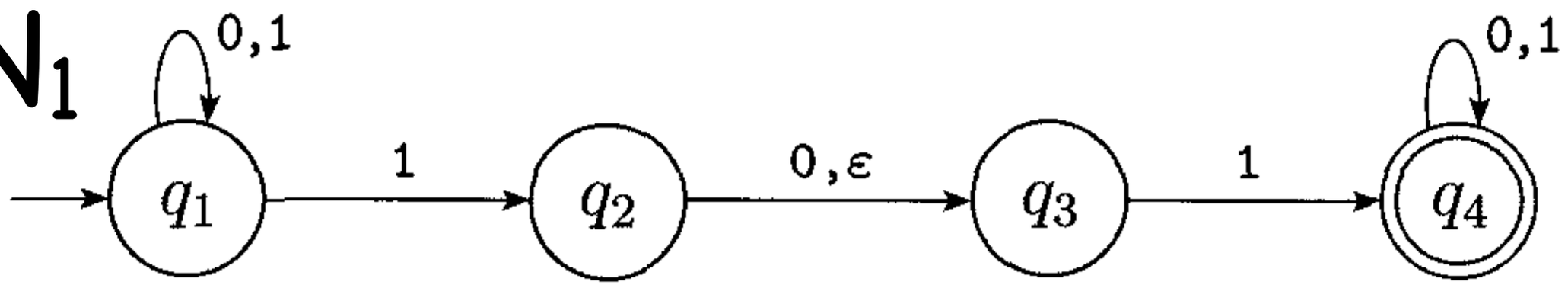


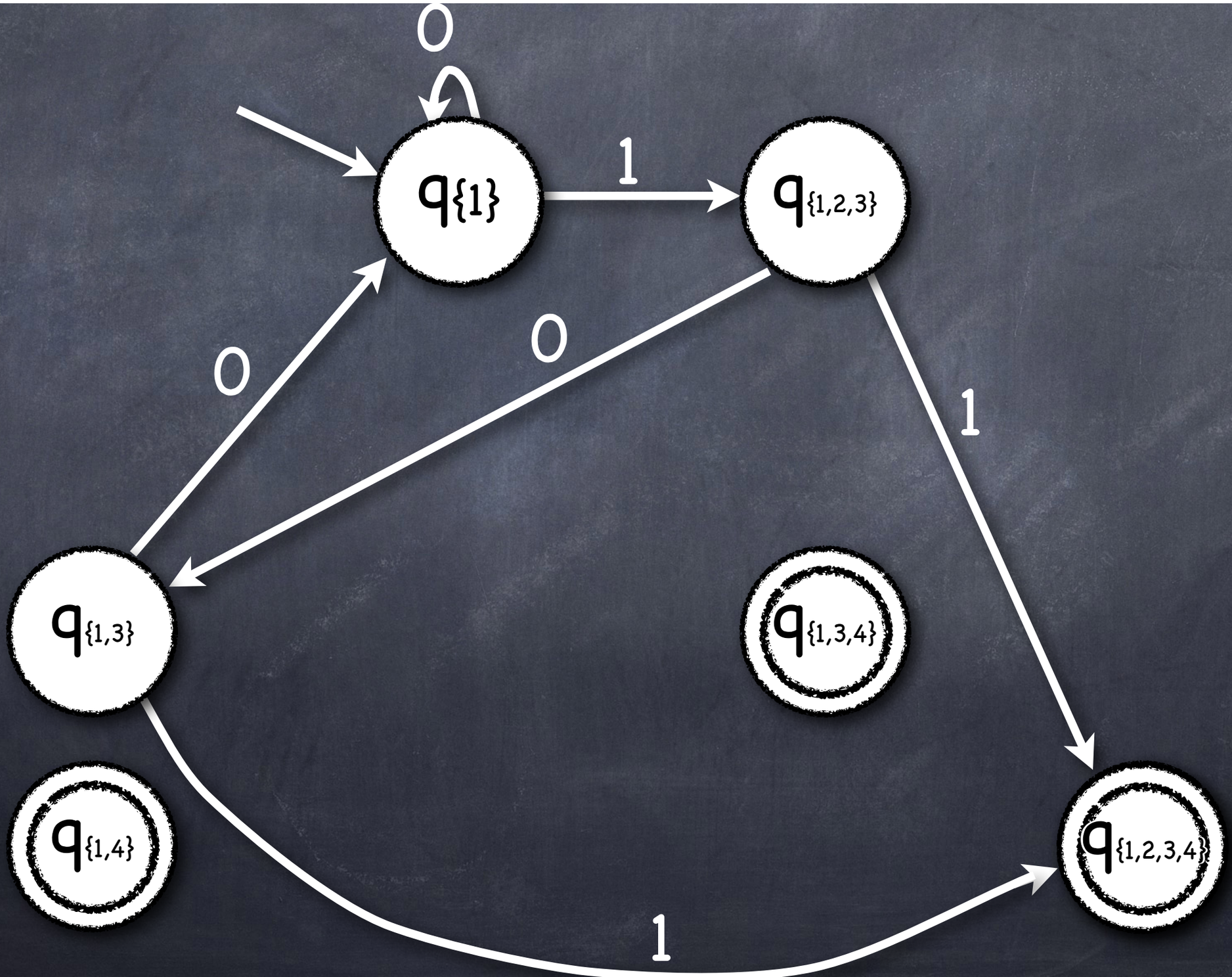
N_1



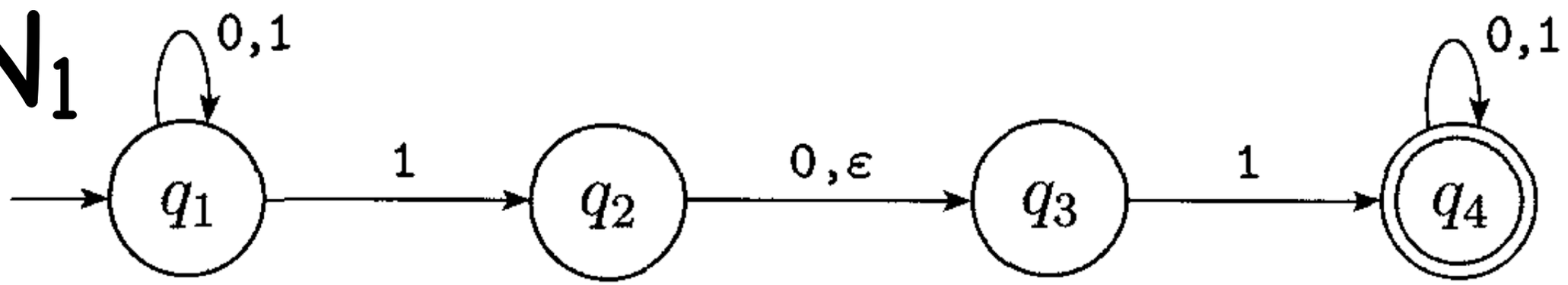


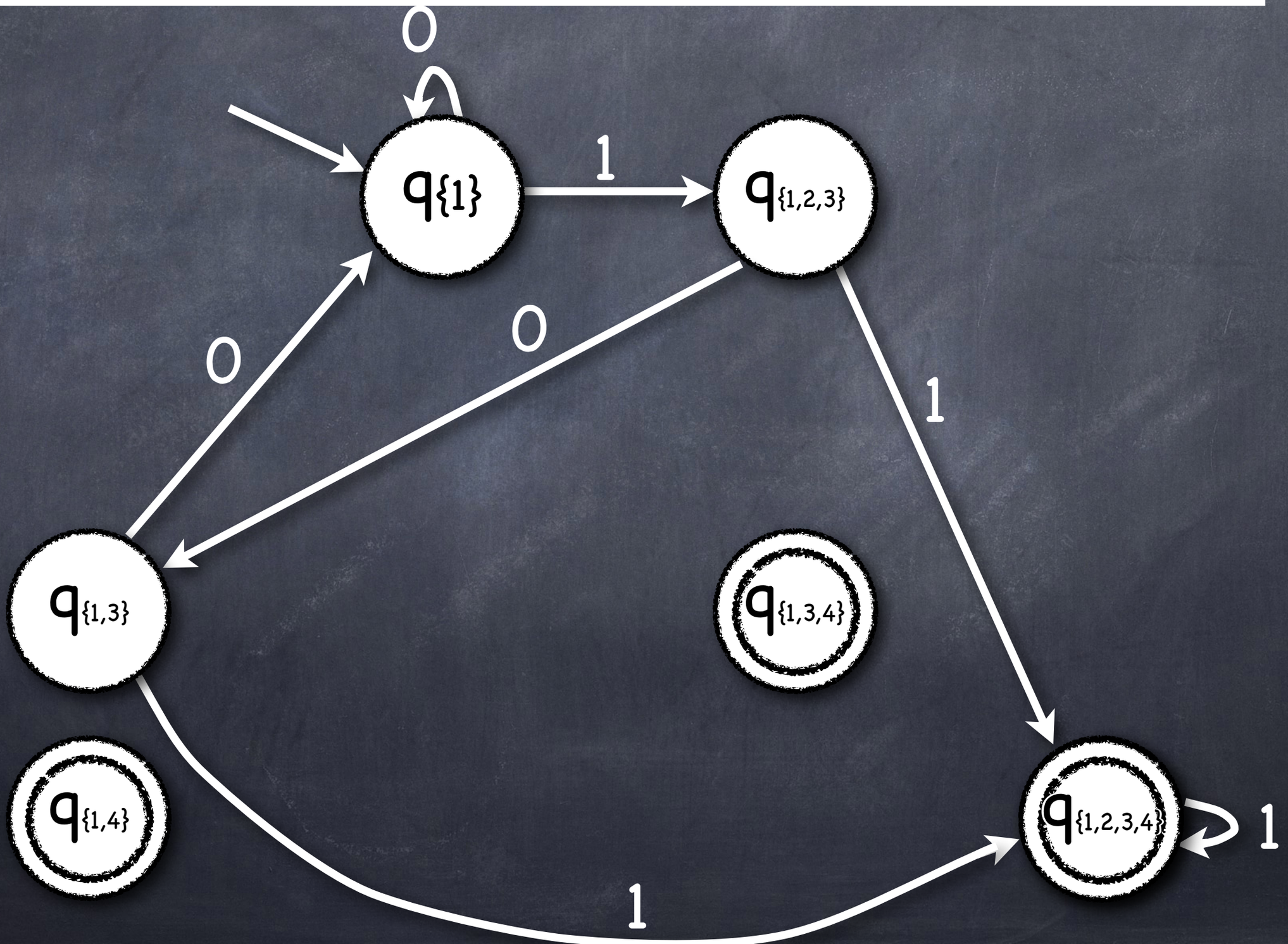
N_1



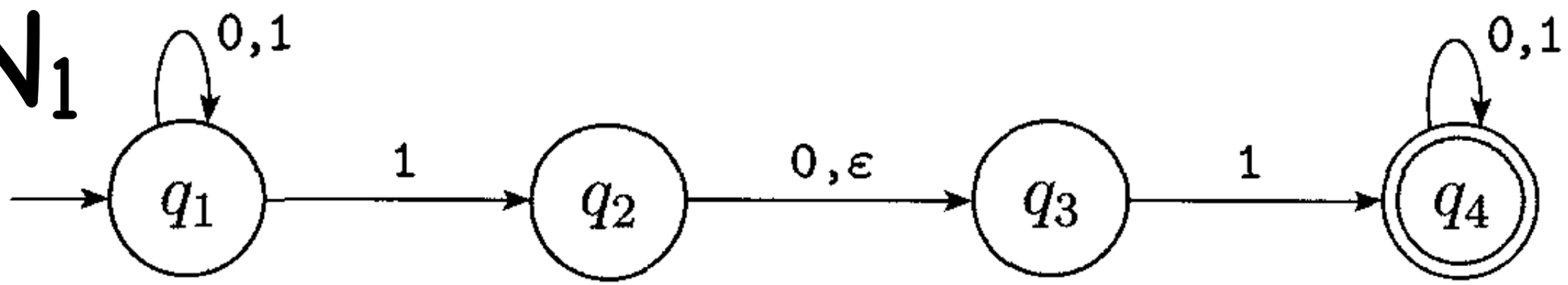


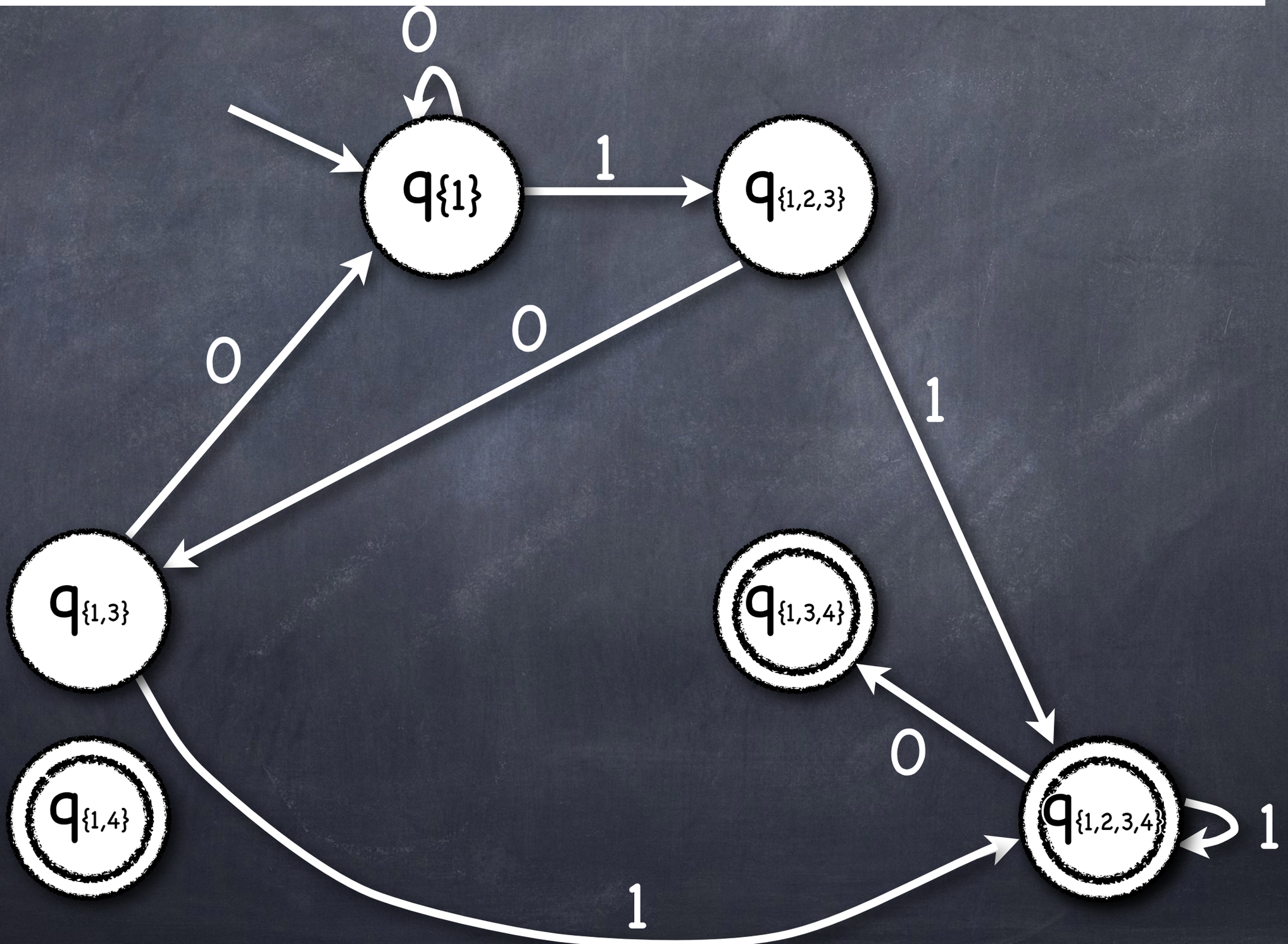
N_1



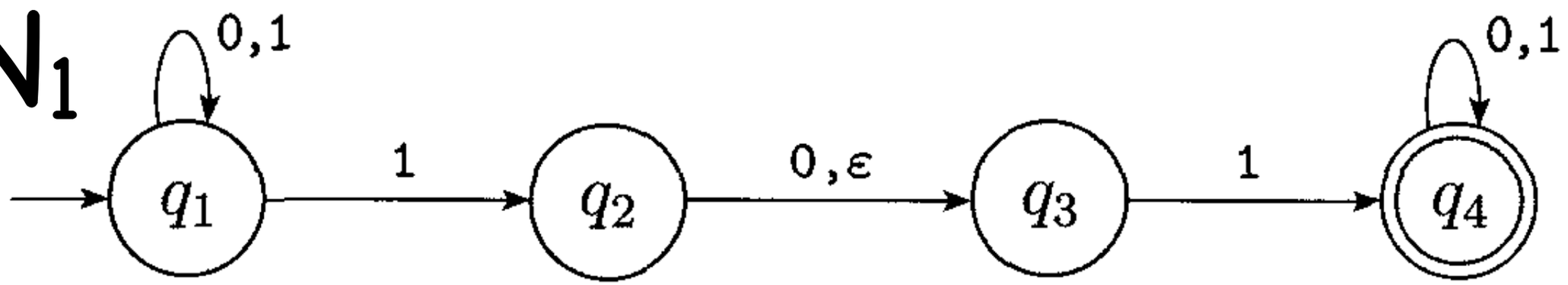


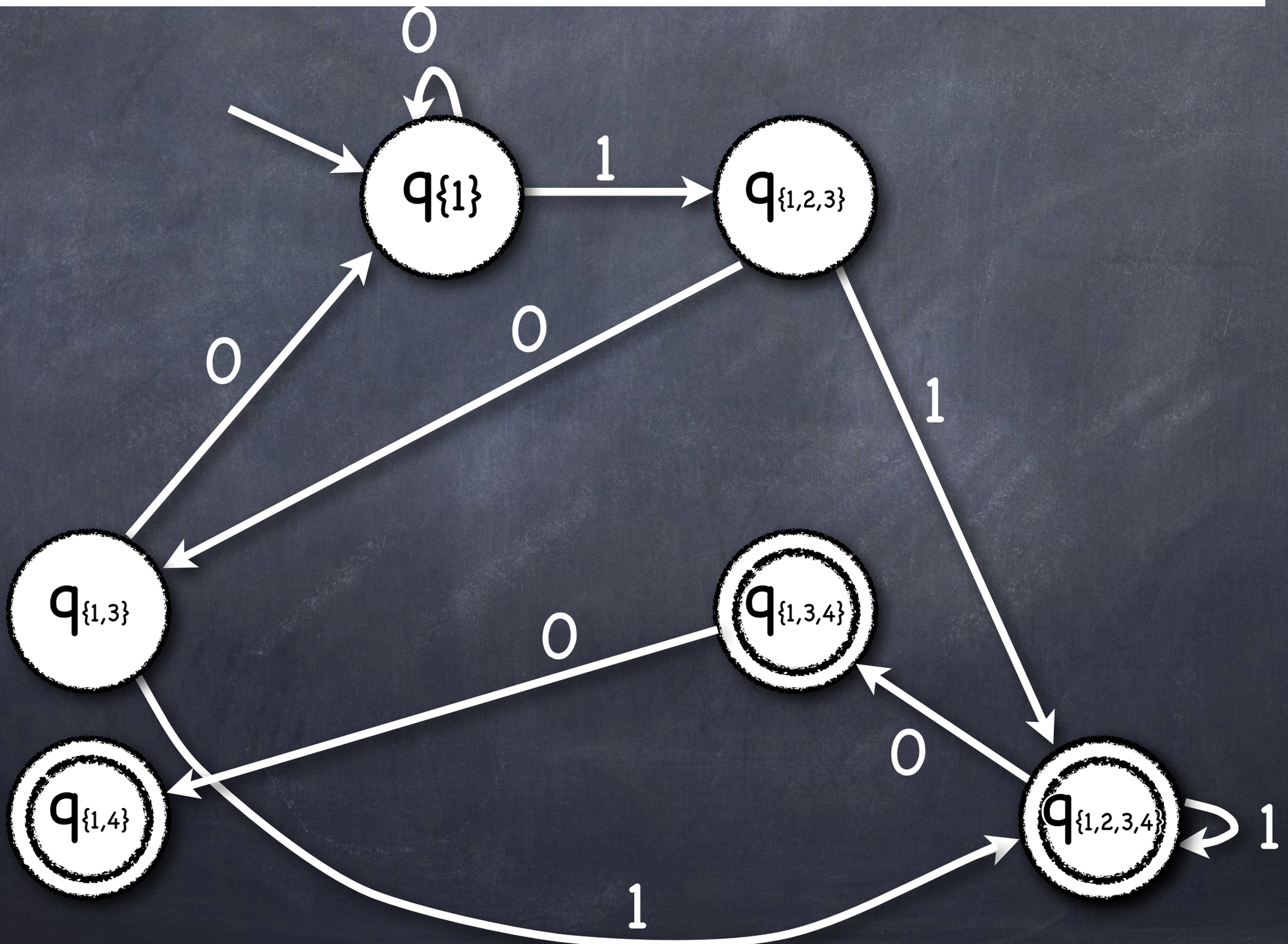
N_1



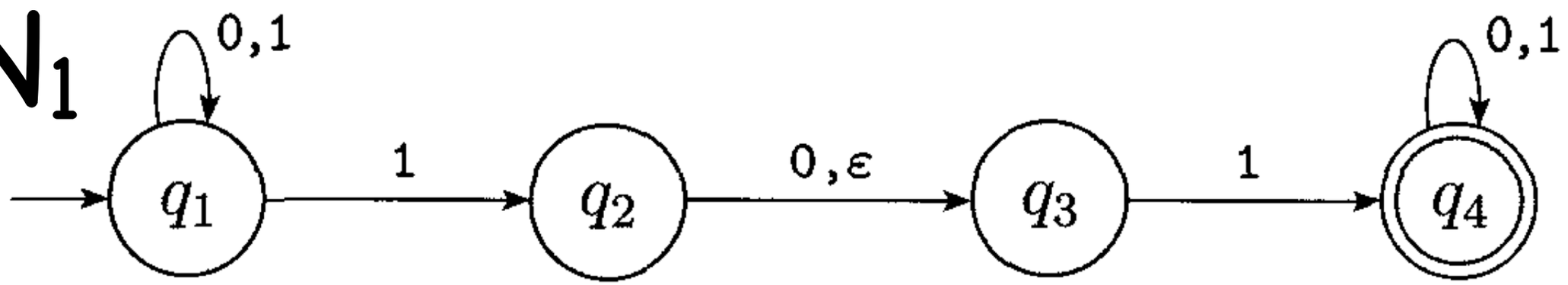


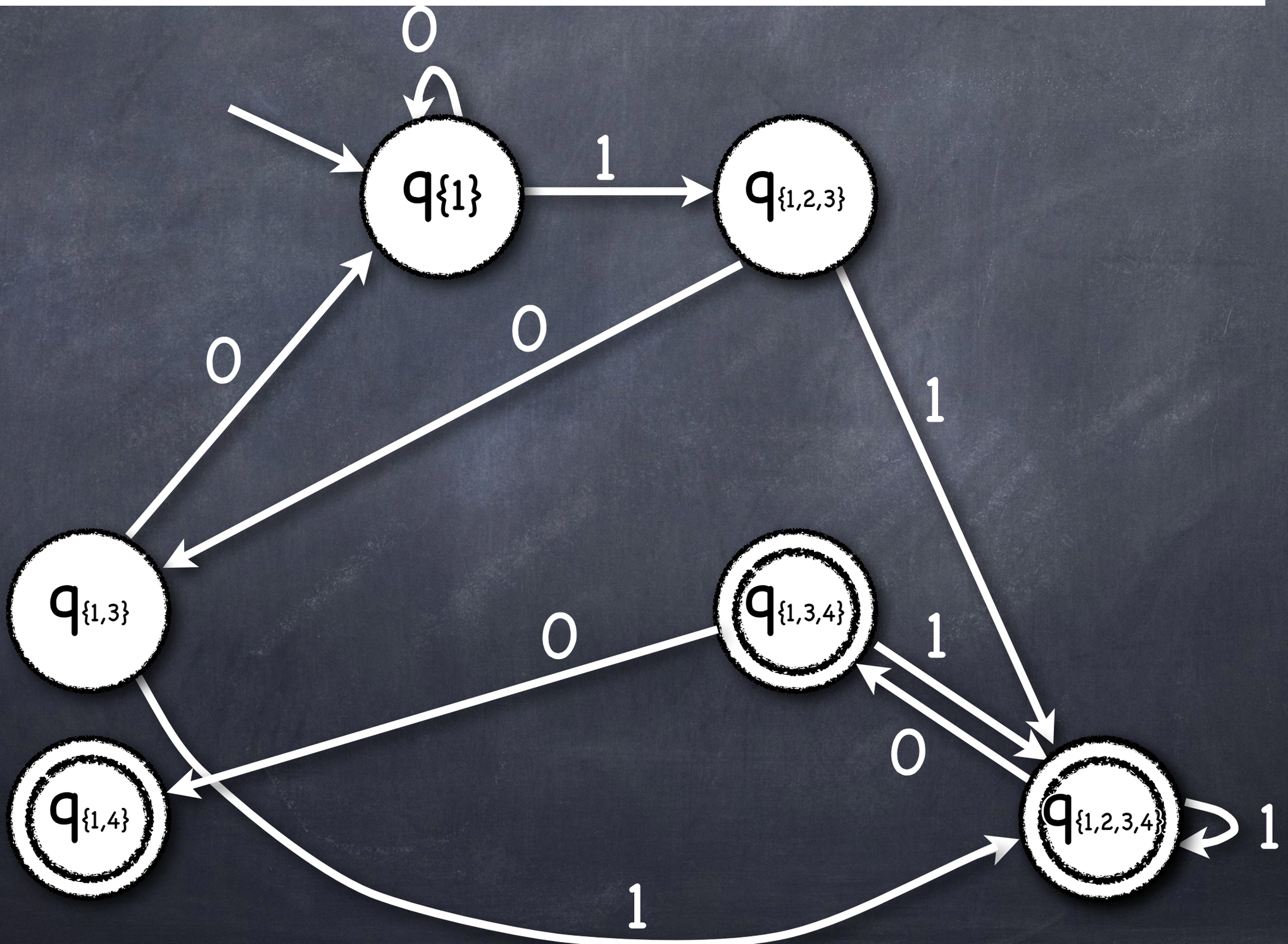
N_1



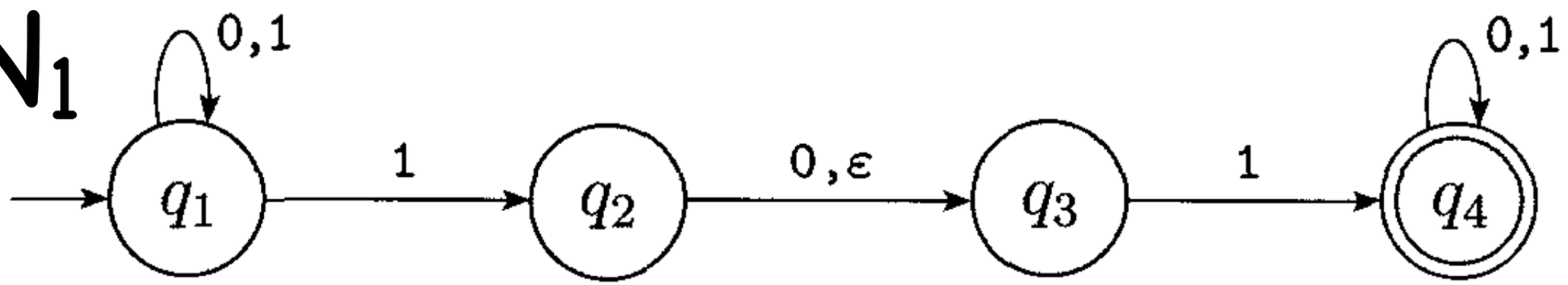


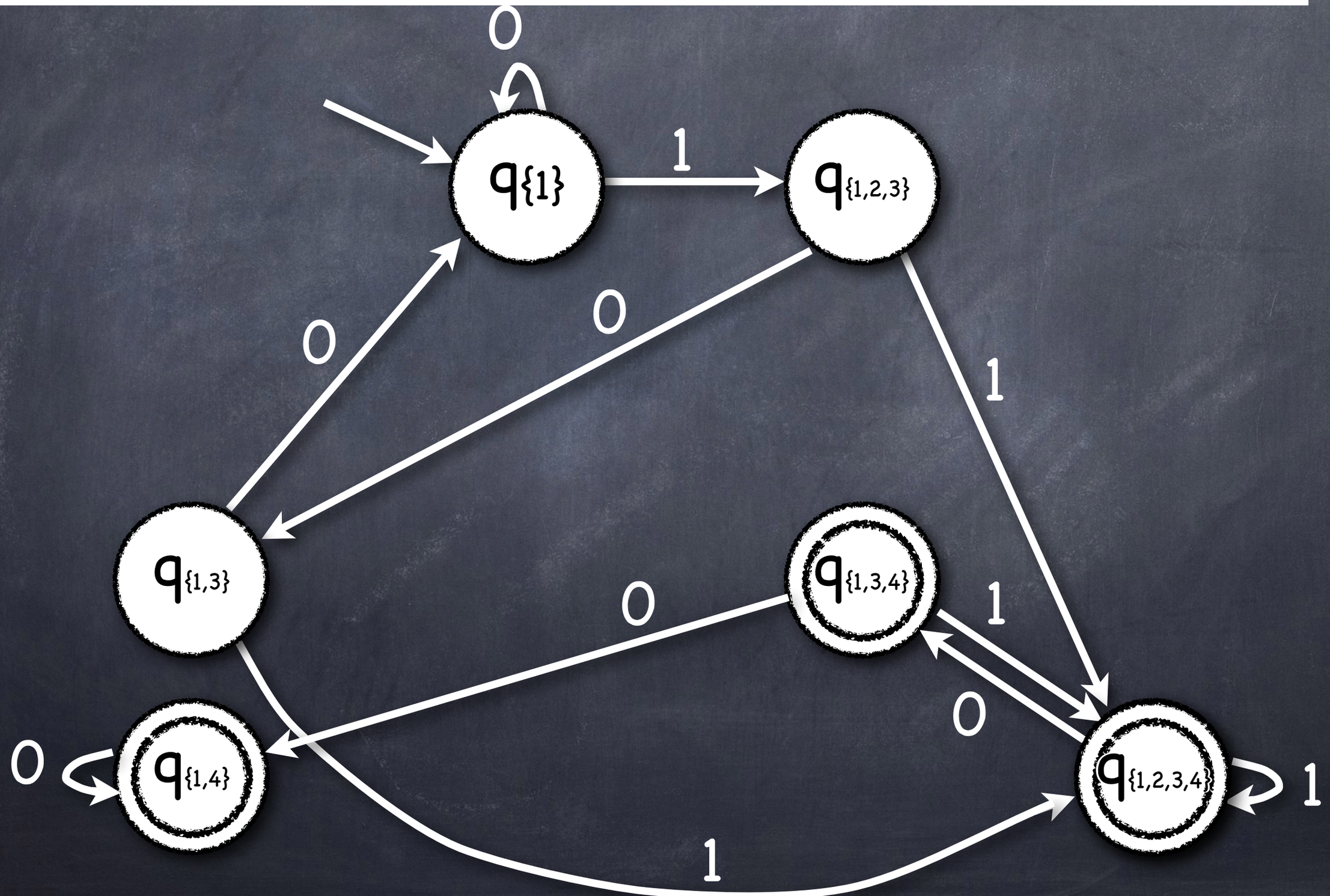
N_1



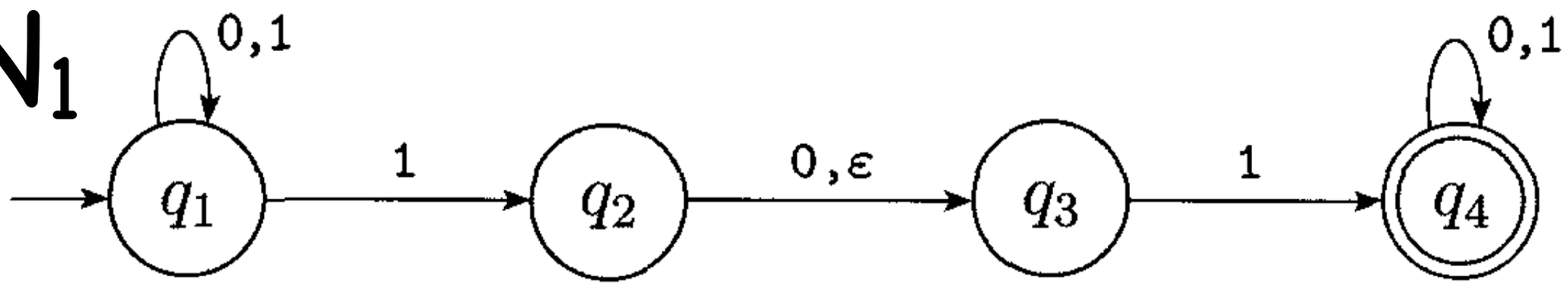


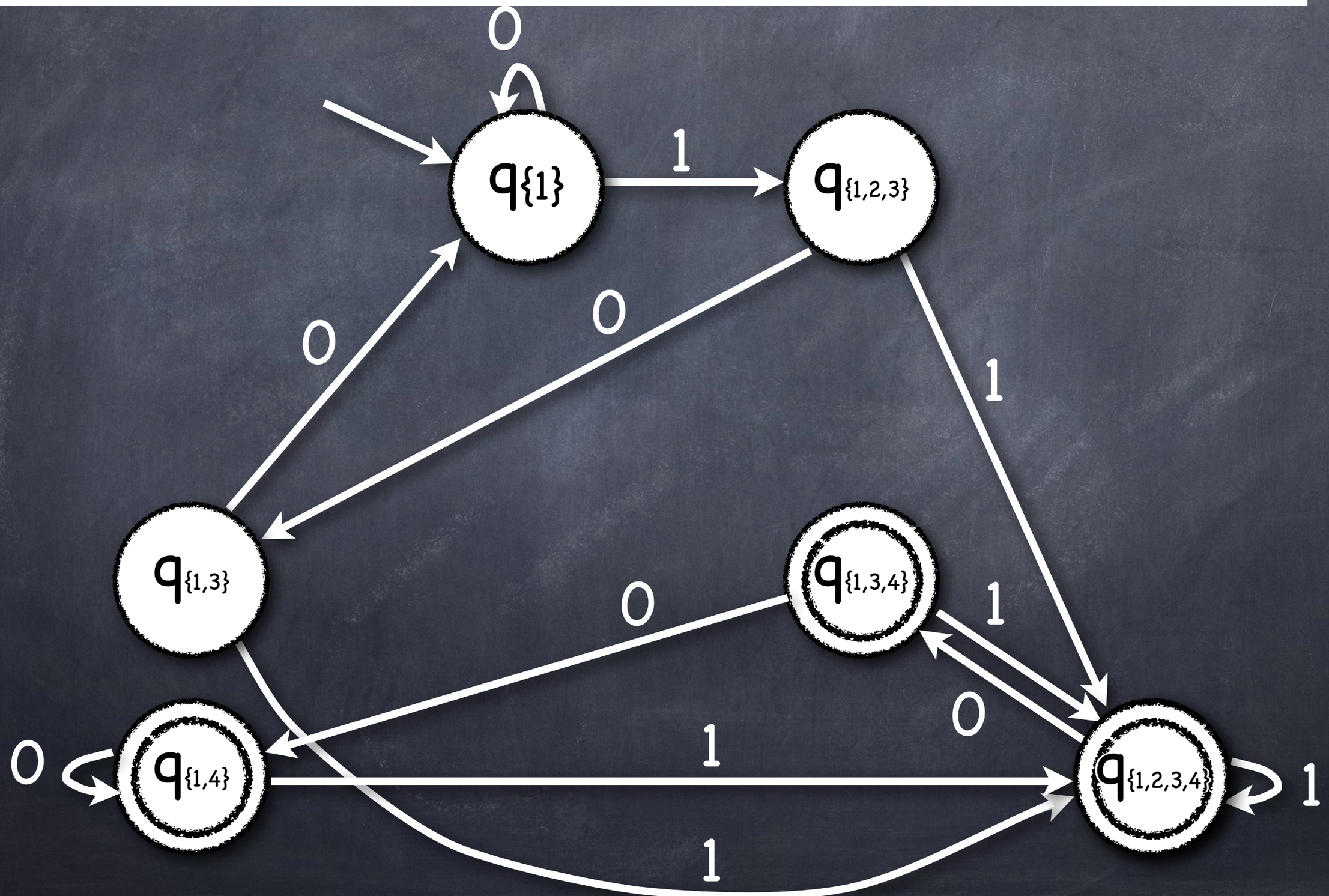
N_1



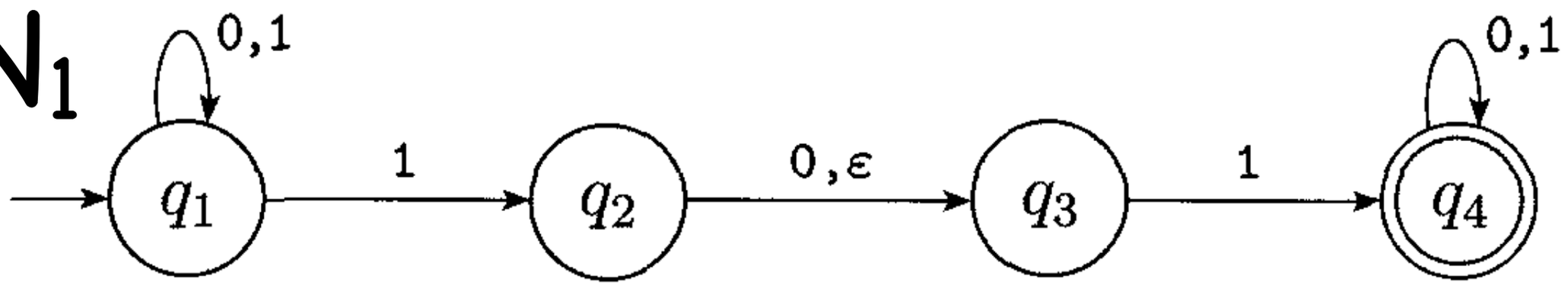


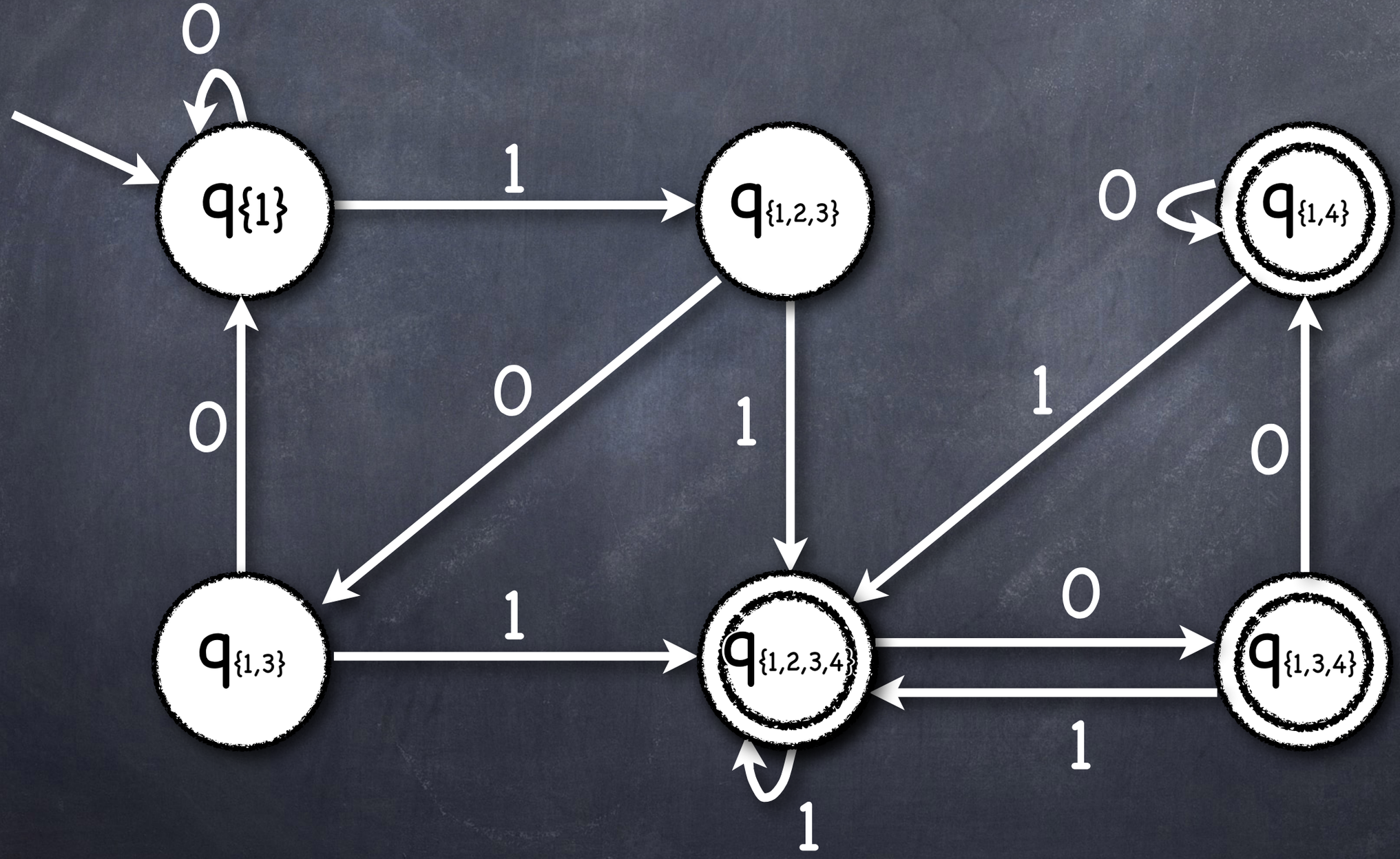
N_1





N_1





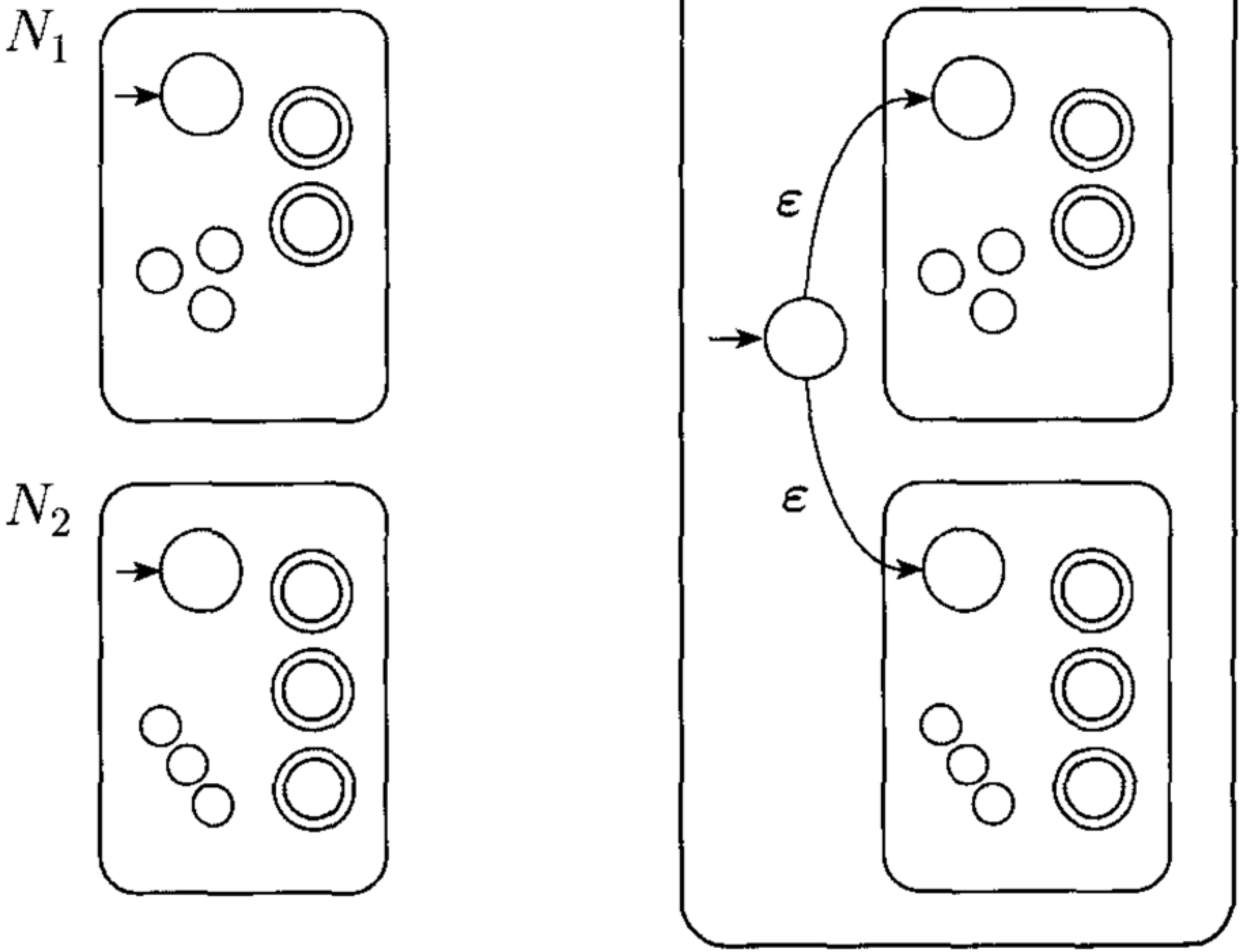
Regular Operations : Kleene's theorem (NFA)

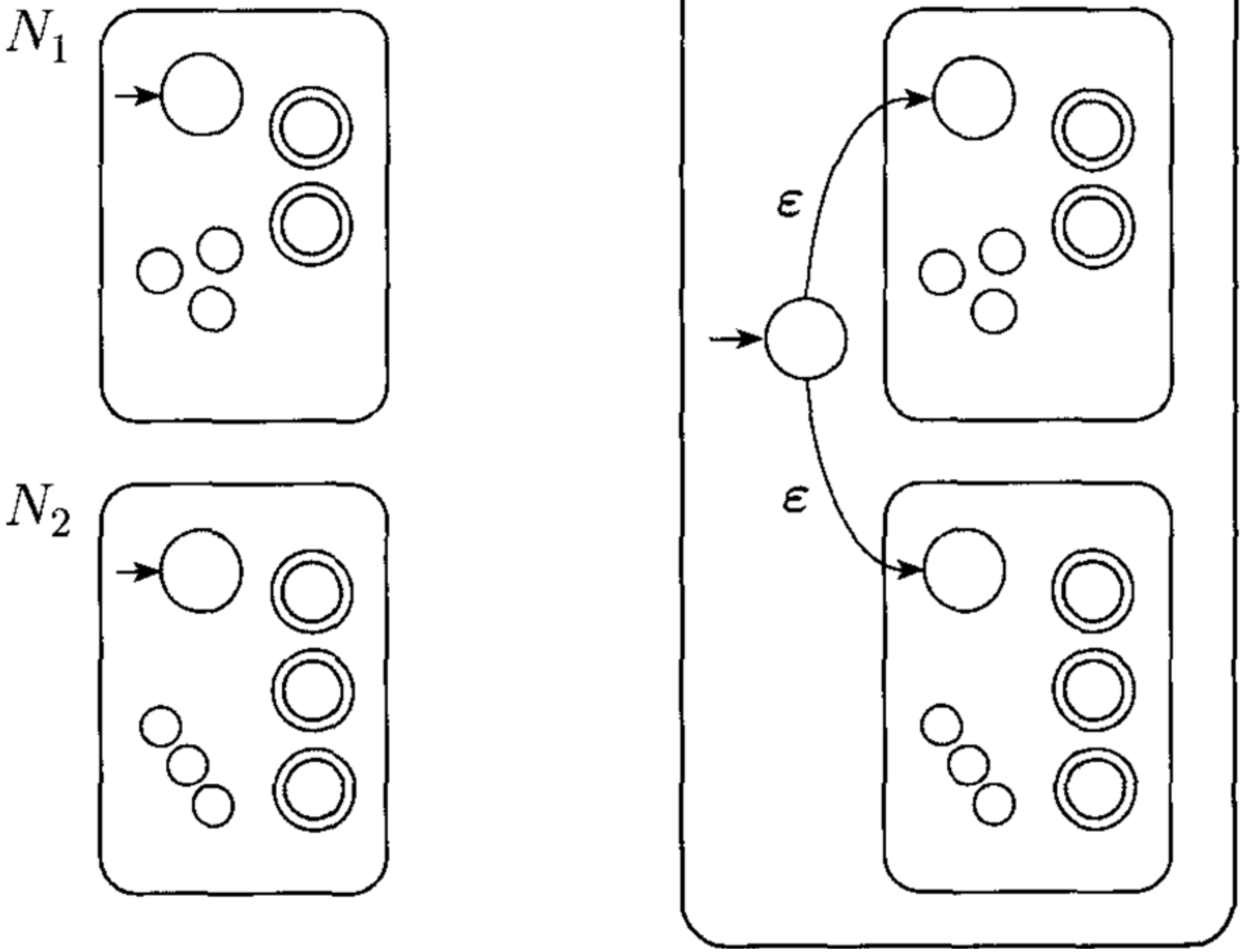
Regular Operations : Kleene's theorem

Regular Operations : Kleene's theorem

THEOREM 1.45

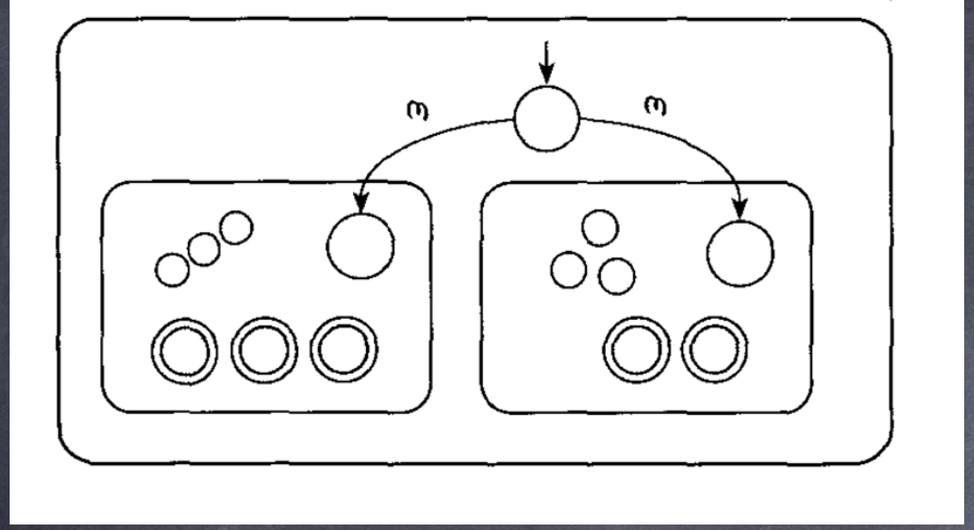
The class of regular languages is closed under the union operation.


N_1
N_2
N
ε
ε

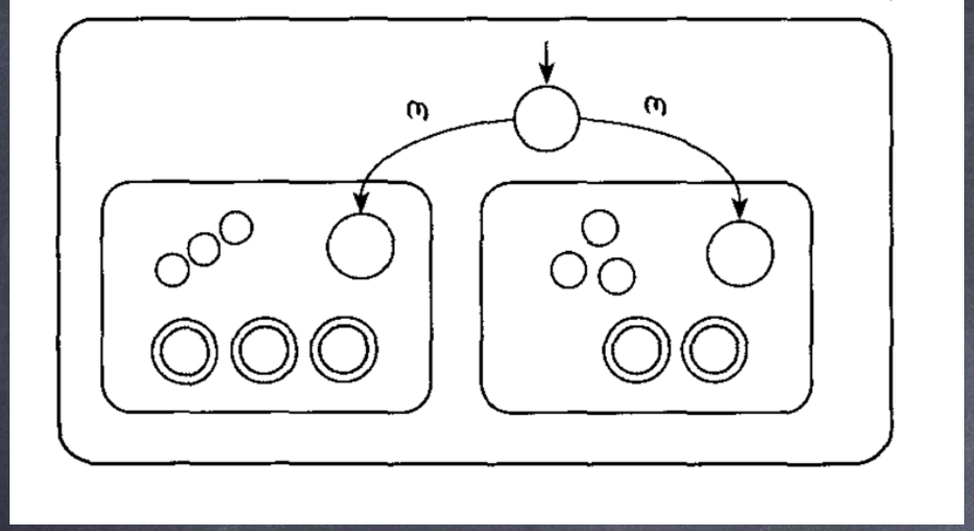


THEOREM 1.45

The class of regular languages is closed under the union operation.



Kleene's theorem



Kleene's theorem

- Let $N_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a NFA accepting L_A and $N_B=(Q_B,\Sigma,\delta_B,q_{0B},F_B)$ be a NFA accepting L_B ($Q_A \cap Q_B=\varnothing$).
- Consider $N_U=(\{q_0\} \cup Q_A \cup Q_B, \Sigma, \delta_U, q_0, F_U)$ where

  $\delta_U(q_0,\varepsilon) = \{q_{0A},q_{0B}\}$, $\delta_U(q_0,a) = \varnothing$ for all $a \neq \varepsilon$,

  $\delta_U(q,a) = \delta_X(q,a)$ for all $q \in Q_X$, $X \in \{A,B\}$, and all a,

  $F_U = F_A \cup F_B$.
- $L_U = L_A \cup L_B$.

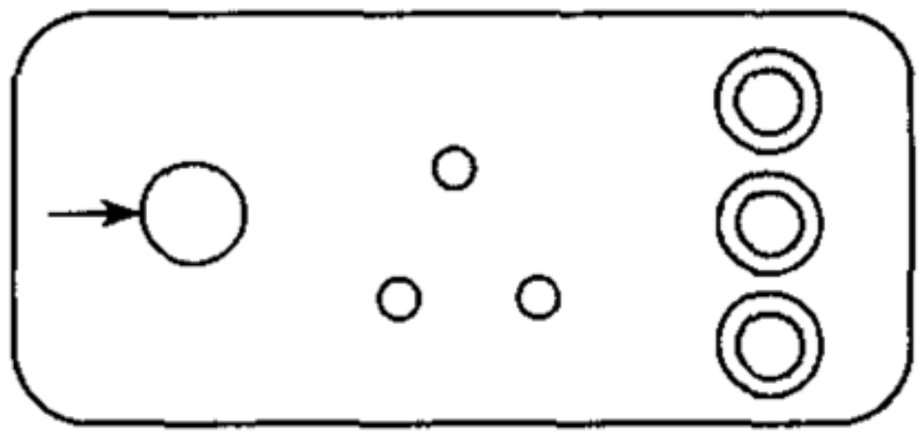
Regular Operations : Kleene's theorem

Regular Operations : Kleene's theorem

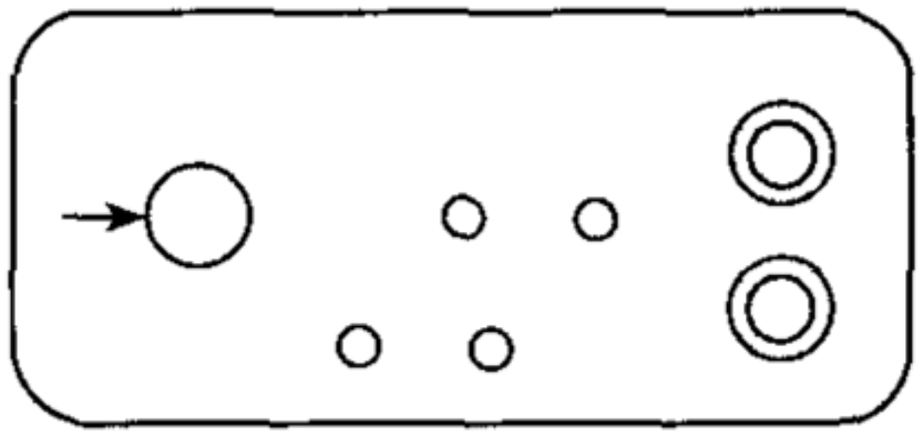
THEOREM 1.47

The class of regular languages is closed under the concatenation operation.

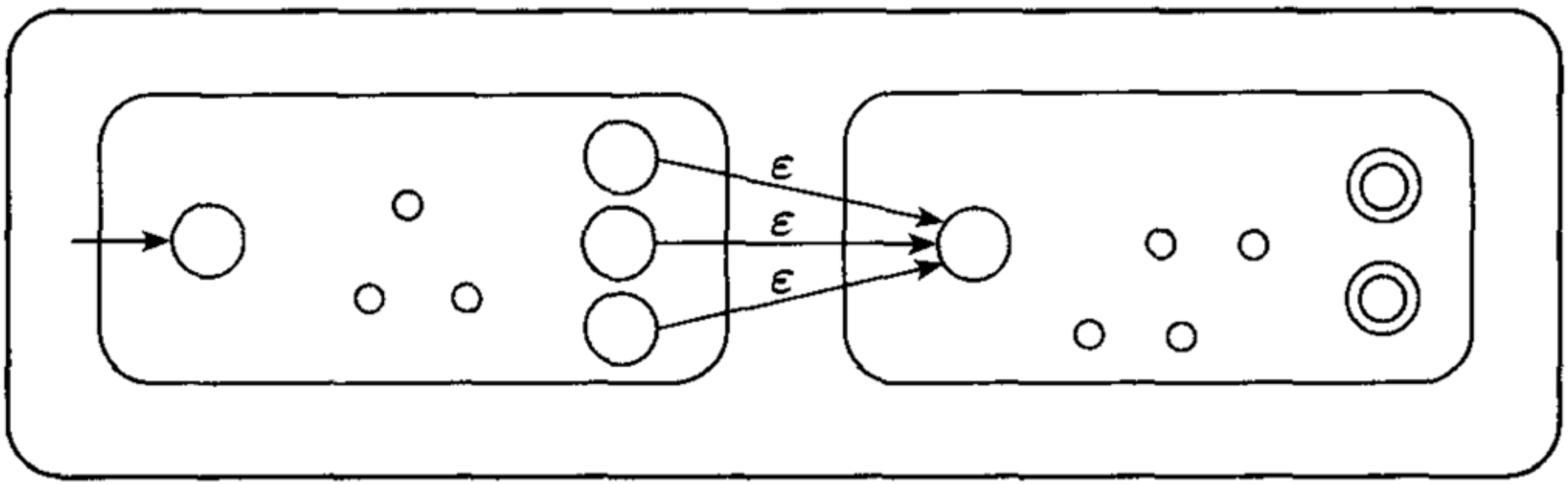
N_1

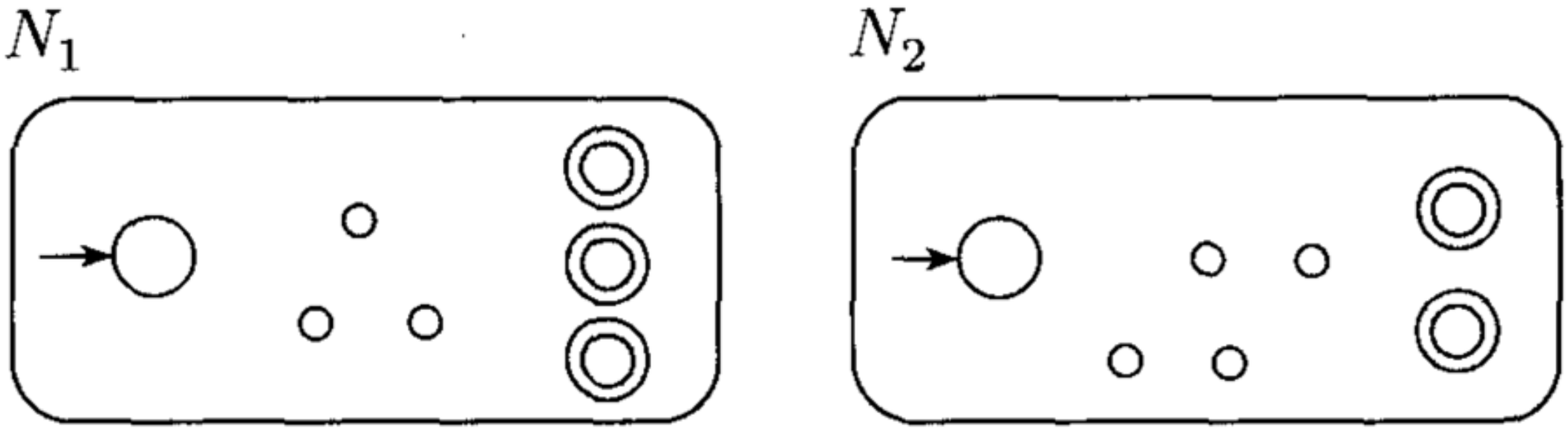


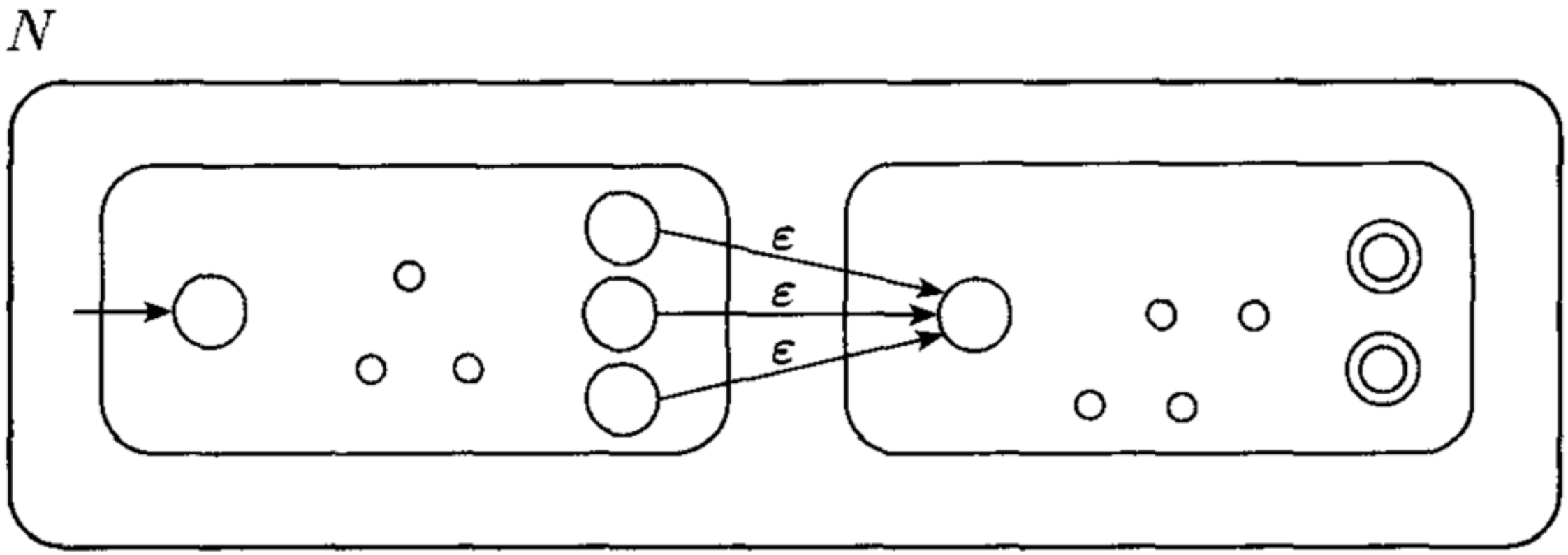
N_2



N

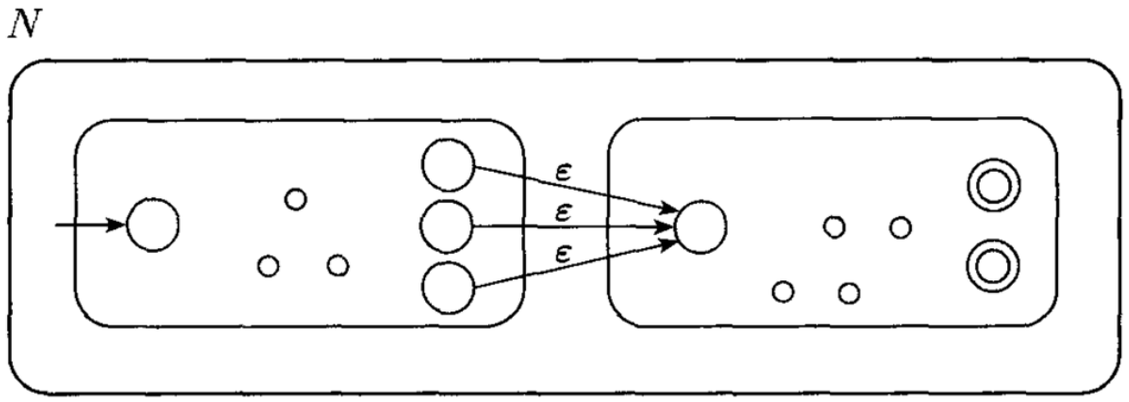




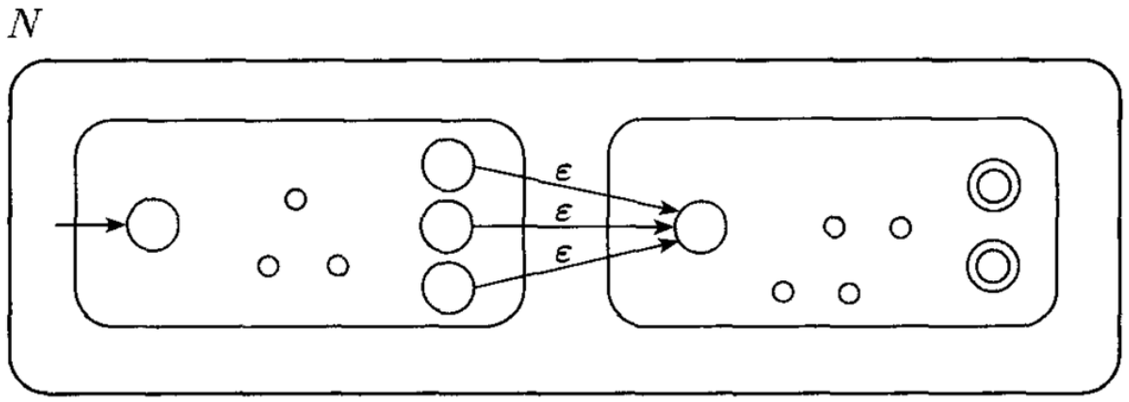


THEOREM 1.47

The class of regular languages is closed under the concatenation operation.



Kleene's theorem



Kleene's theorem

- Let $N_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a NFA accepting L_A and $N_B=(Q_B,\Sigma,\delta_B,q_{0B},F_B)$ be a NFA accepting L_B ($Q_A \cap Q_B=\varnothing$).

- Consider $N_C=(\ Q_A \cup Q_B\ ,\Sigma,\delta_C,q_{0A},F_B)$ where

  $\delta_C(q,a) = \delta_B(q,a)$ for all $q\in Q_B$, all a,

  $\delta_C(q,a) = \delta_A(q,a)$ for all $q\in Q_A$, all $a\neq\varepsilon$,

  $\delta_C(q,\varepsilon) = \delta_A(q,\varepsilon)$ for all $q\in Q_A\setminus F_A$,

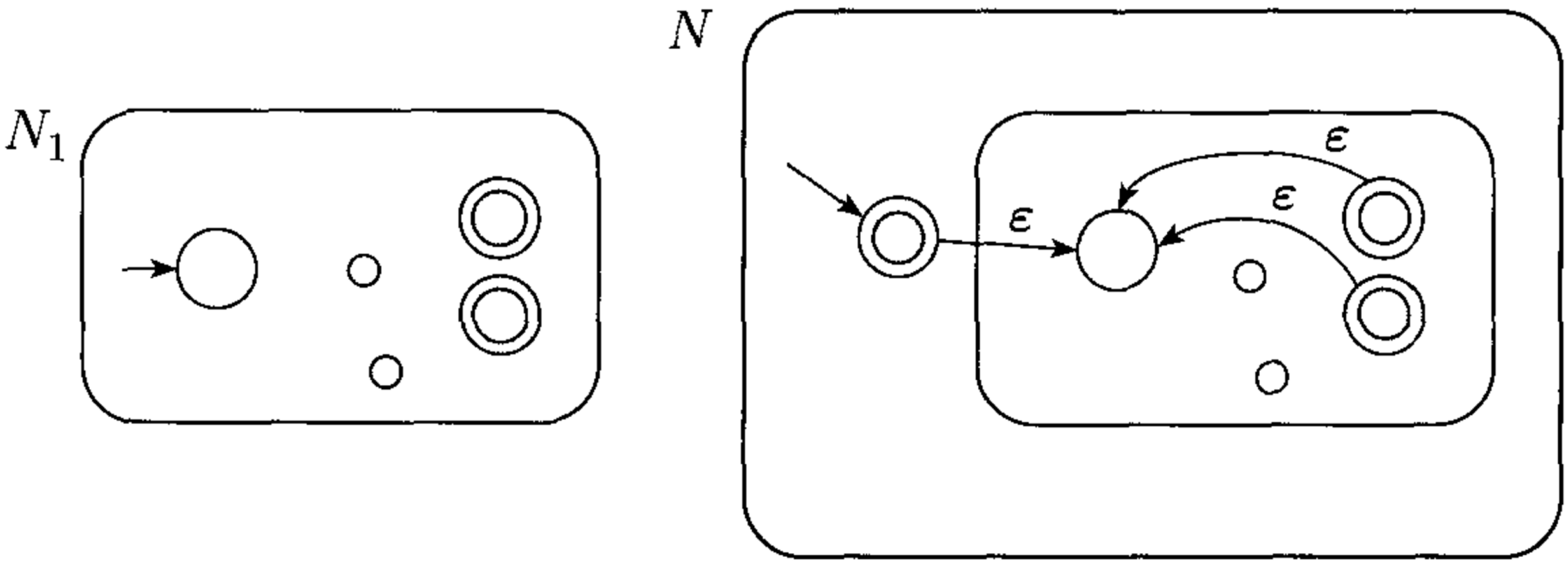
  $\delta_C(q,\varepsilon) = \delta_A(q,\varepsilon)\cup\{q_{0B}\}$ for all $q\in F_A$.

- $L_C = L_A \circ L_B$.

Regular Operations : Kleene's theorem

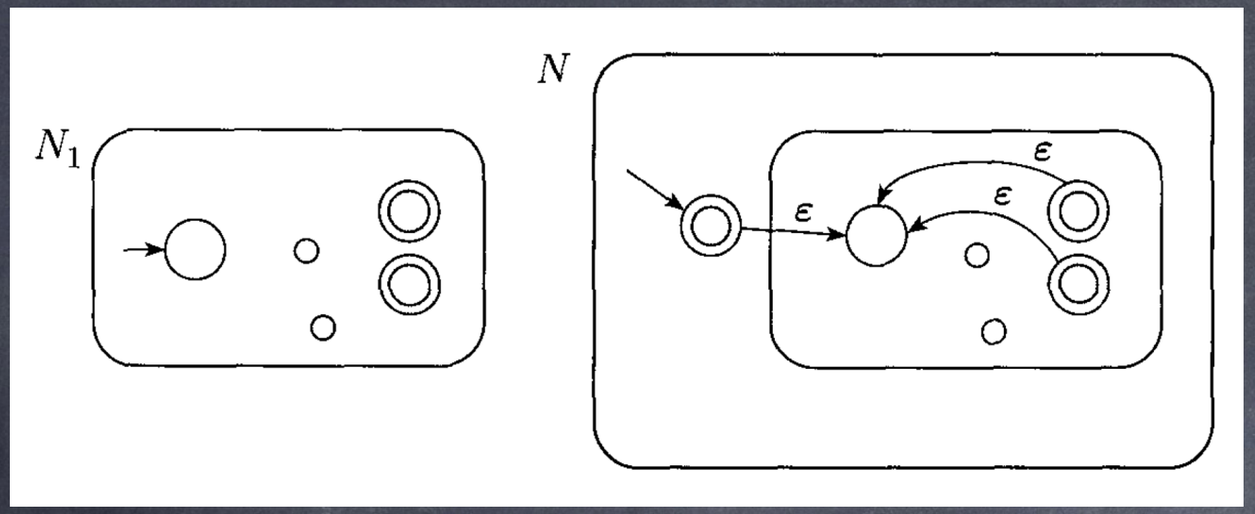
THEOREM 1.49

The class of regular languages is closed under the star operation.

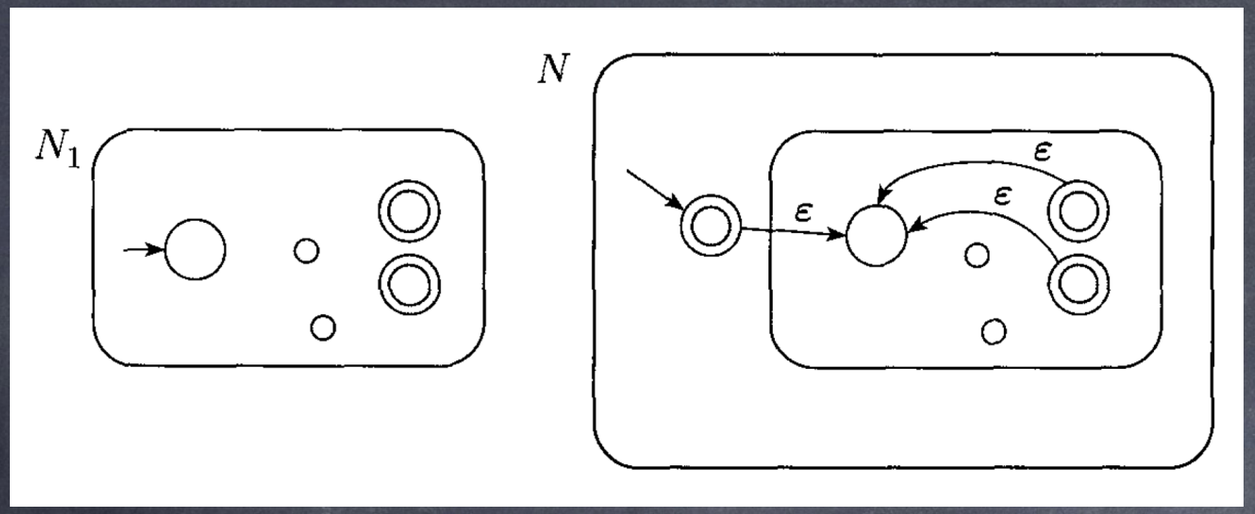


THEOREM 1.49

The class of regular languages is closed under the star operation.



Kleene's theorem

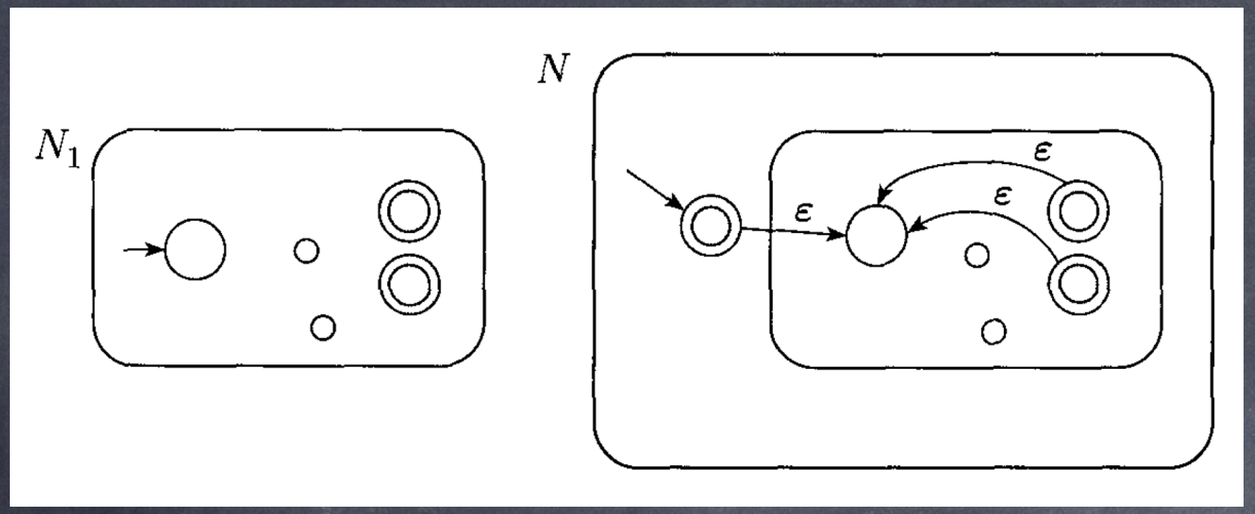


Kleene's theorem

- Let $N_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a NFA accepting L_A.

Kleene's theorem

- Let $N_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a NFA accepting L_A.



Kleene's theorem

- Let $N_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a NFA accepting L_A.
- Consider $N_S=(Q_A \cup \{q_0\} ,\Sigma,\delta_S,q_0,F_A \cup \{q_0\})$ where

  $\delta_S(q_0,\boldsymbol{\varepsilon}) = q_{0A}$, and $\delta_S(q_0,a) = \varnothing$ for all $a \neq \boldsymbol{\varepsilon}$,

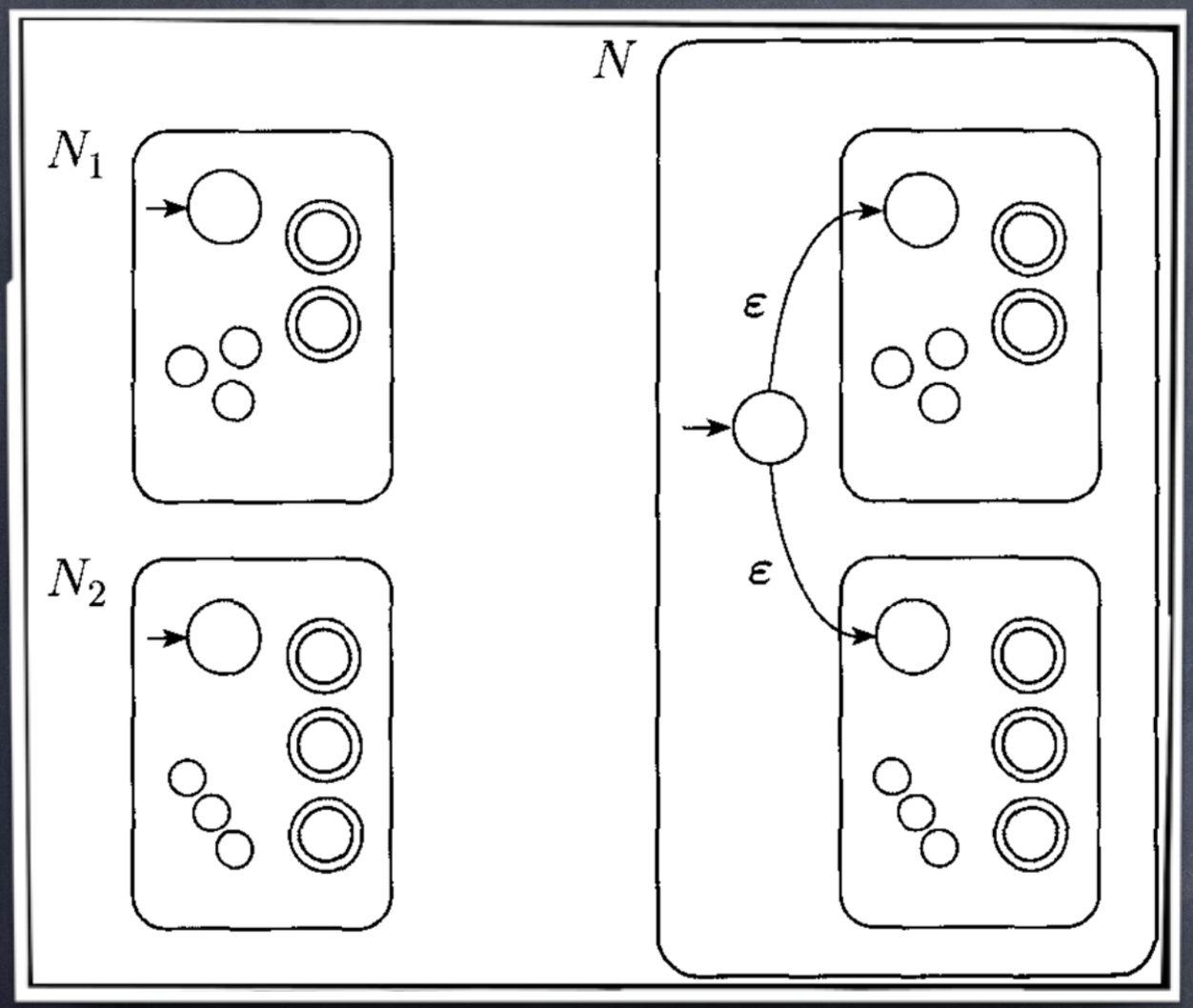
  $\delta_S(q,a) = \delta_A(q,a)$ for all $q \in Q_A \setminus F_A$, all a,

  $\delta_S(q,\boldsymbol{\varepsilon}) = \delta_A(q,\boldsymbol{\varepsilon}) \cup \{q_{0A}\}$ for all $q \in F_A$,

  $\delta_S(q,a) = \delta_A(q,a)$ for all $q \in F_A$, all $a \neq \boldsymbol{\varepsilon}$.
- $L_S = (L_A)^*$.

Regular Operations

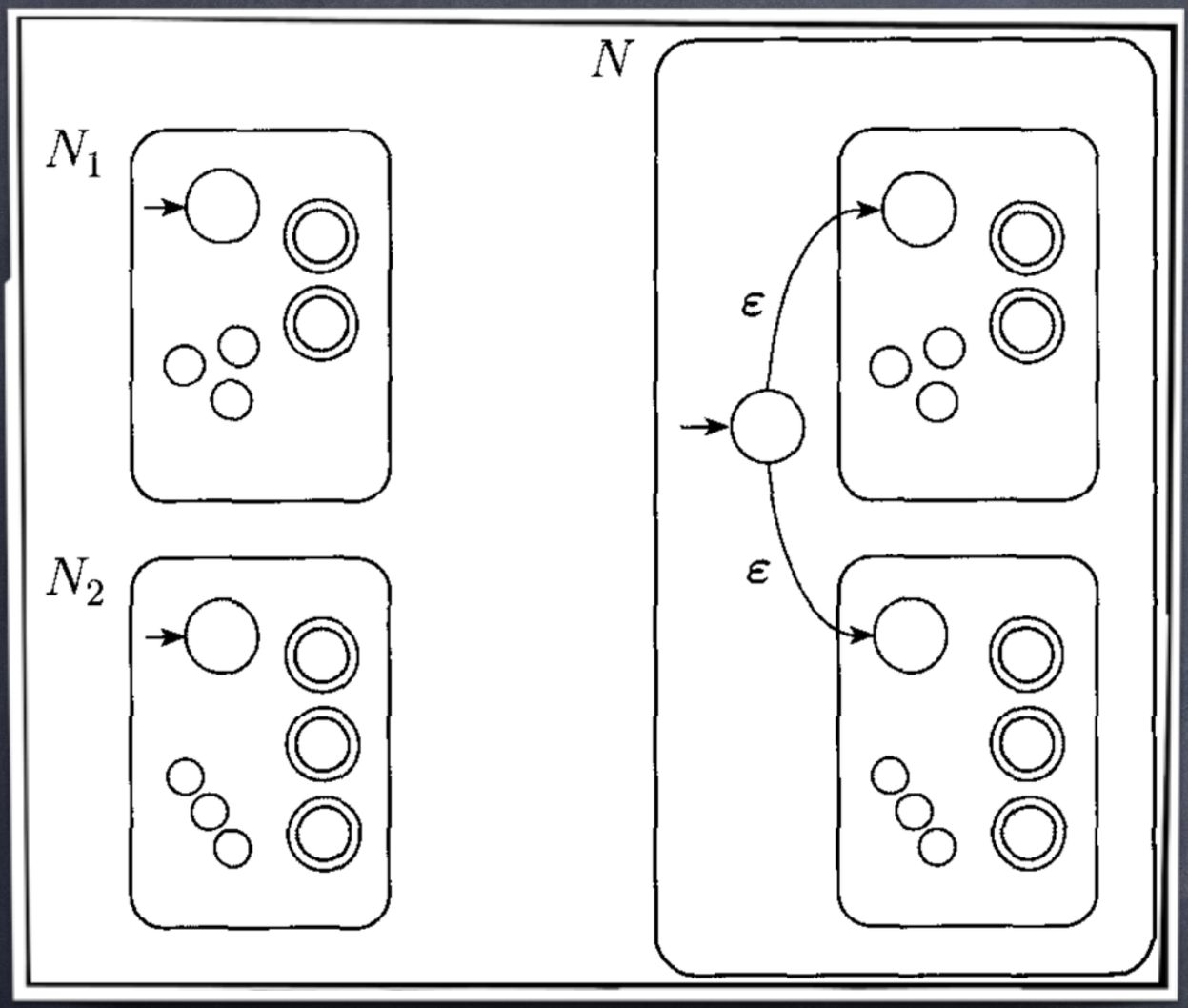
Regular Operations

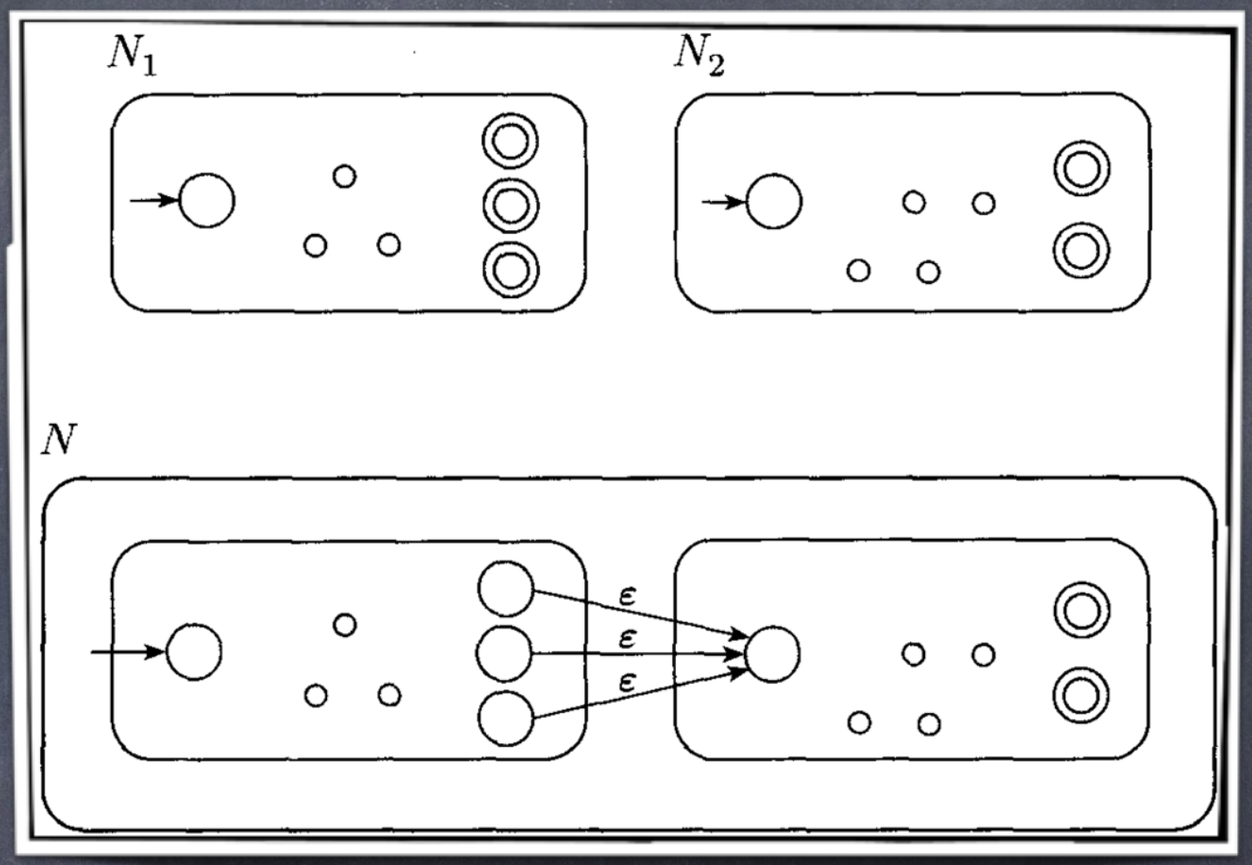


THEOREM 1.45

The class of regular languages is closed under the union operation.

Regular Operations





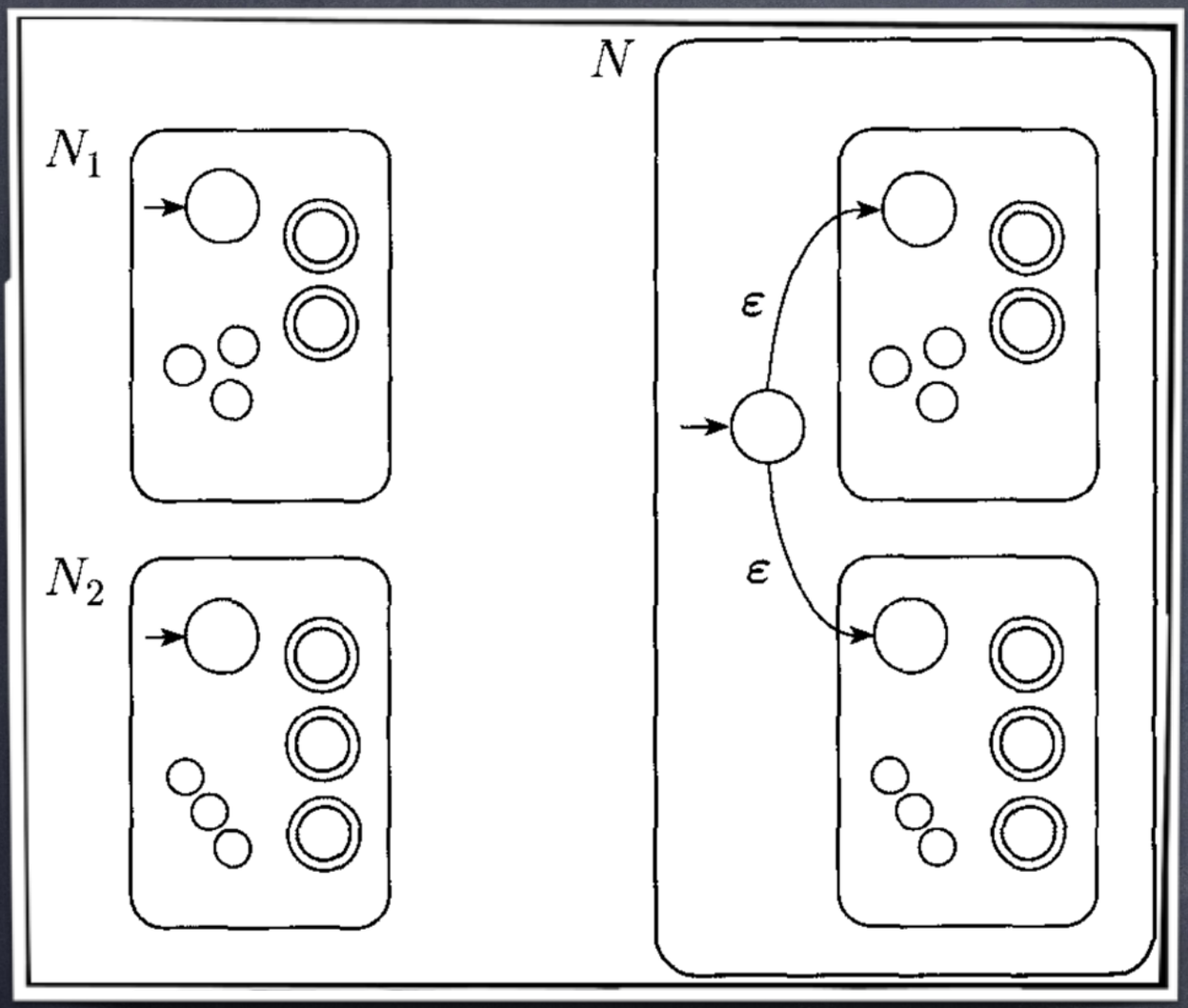
THEOREM 1.45

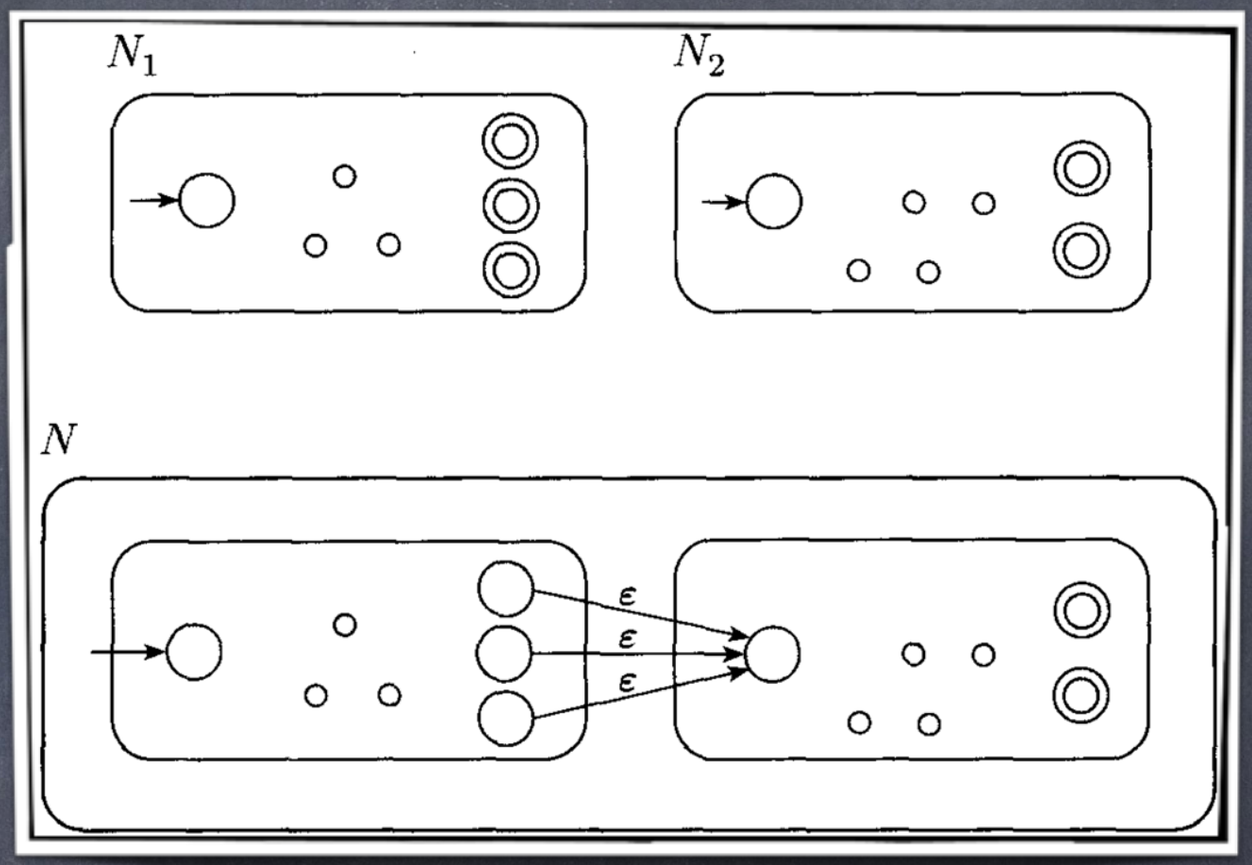
The class of regular languages is closed under the union operation.

THEOREM 1.47

The class of regular languages is closed under the concatenation operation.

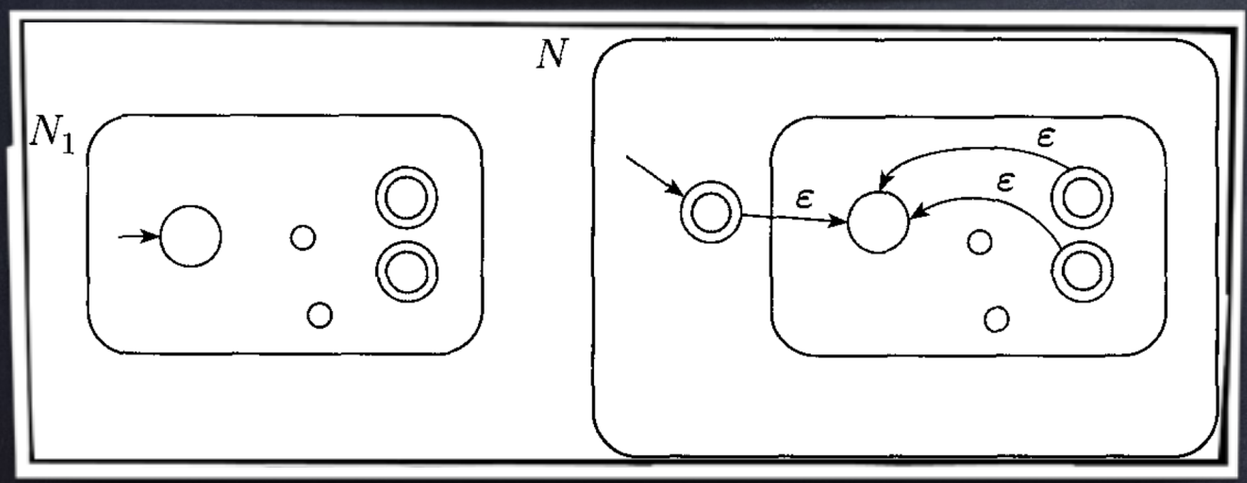
Regular Operations





THEOREM 1.45

The class of regular languages is closed under the union operation.



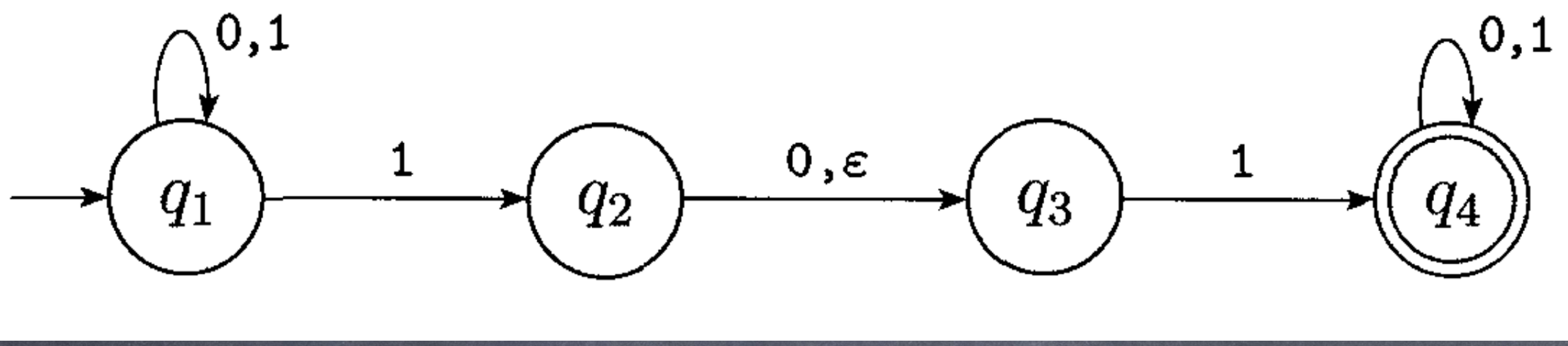
THEOREM 1.47

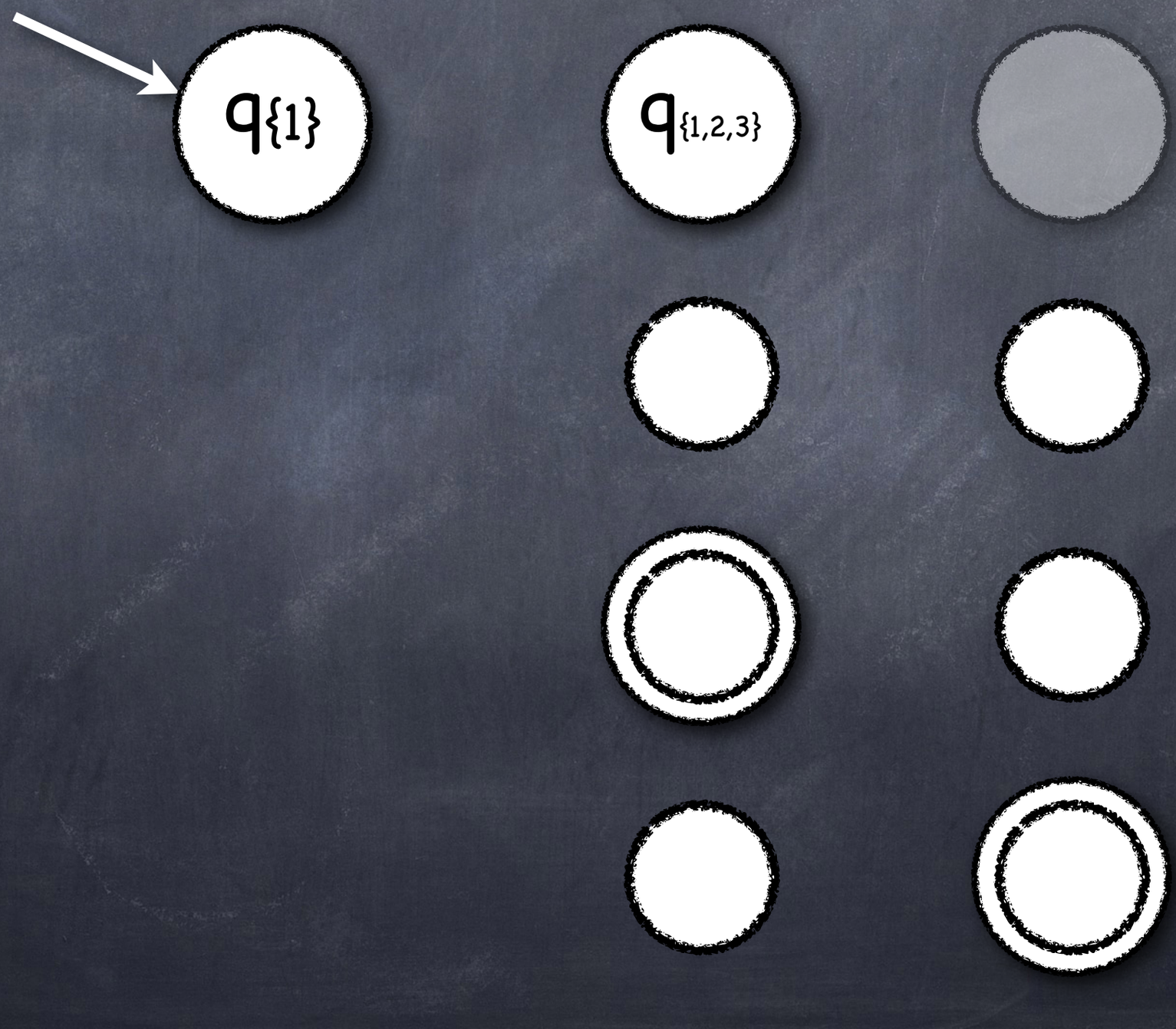
The class of regular languages is closed under the concatenation operation.

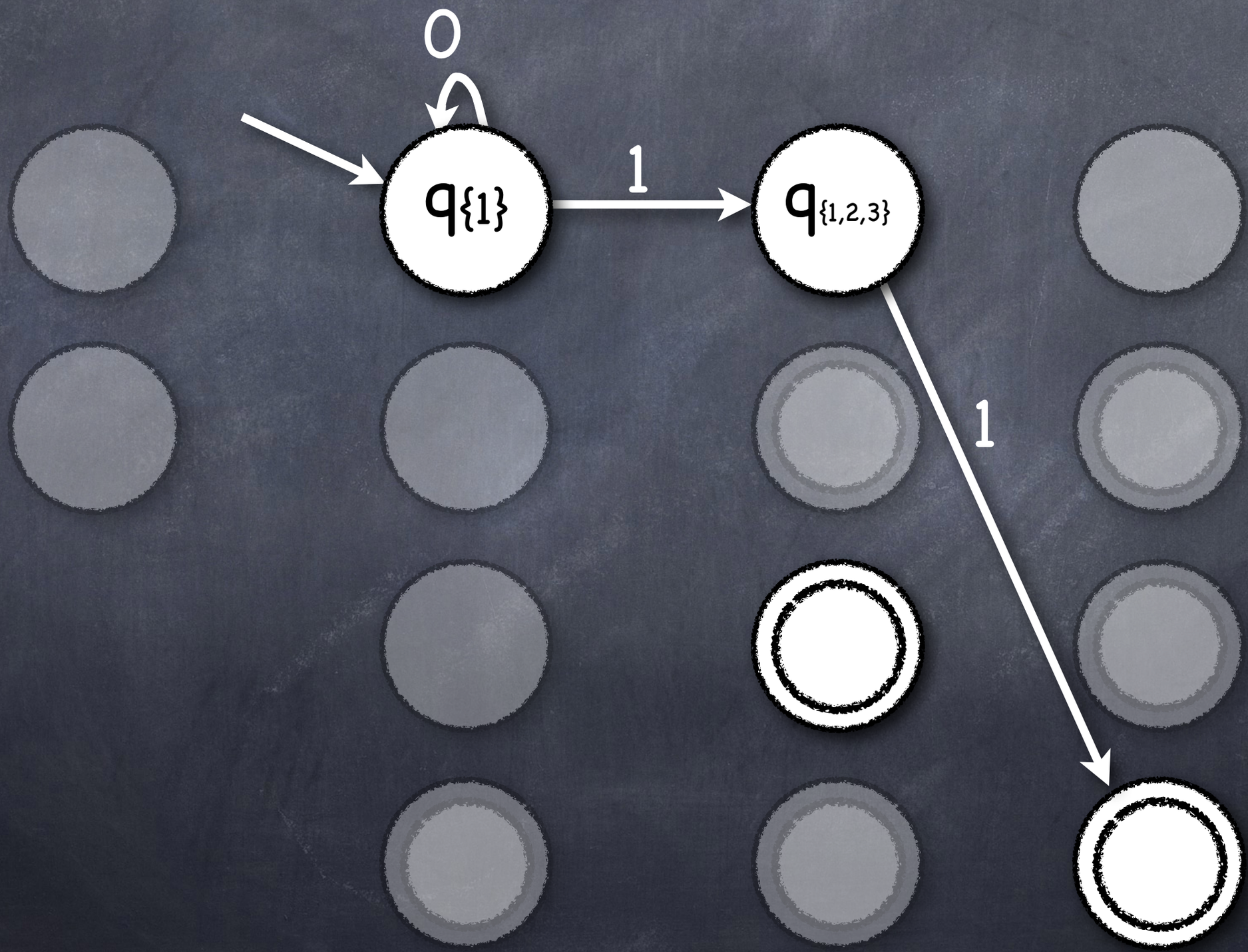
THEOREM 1.49

The class of regular languages is closed under the star operation.

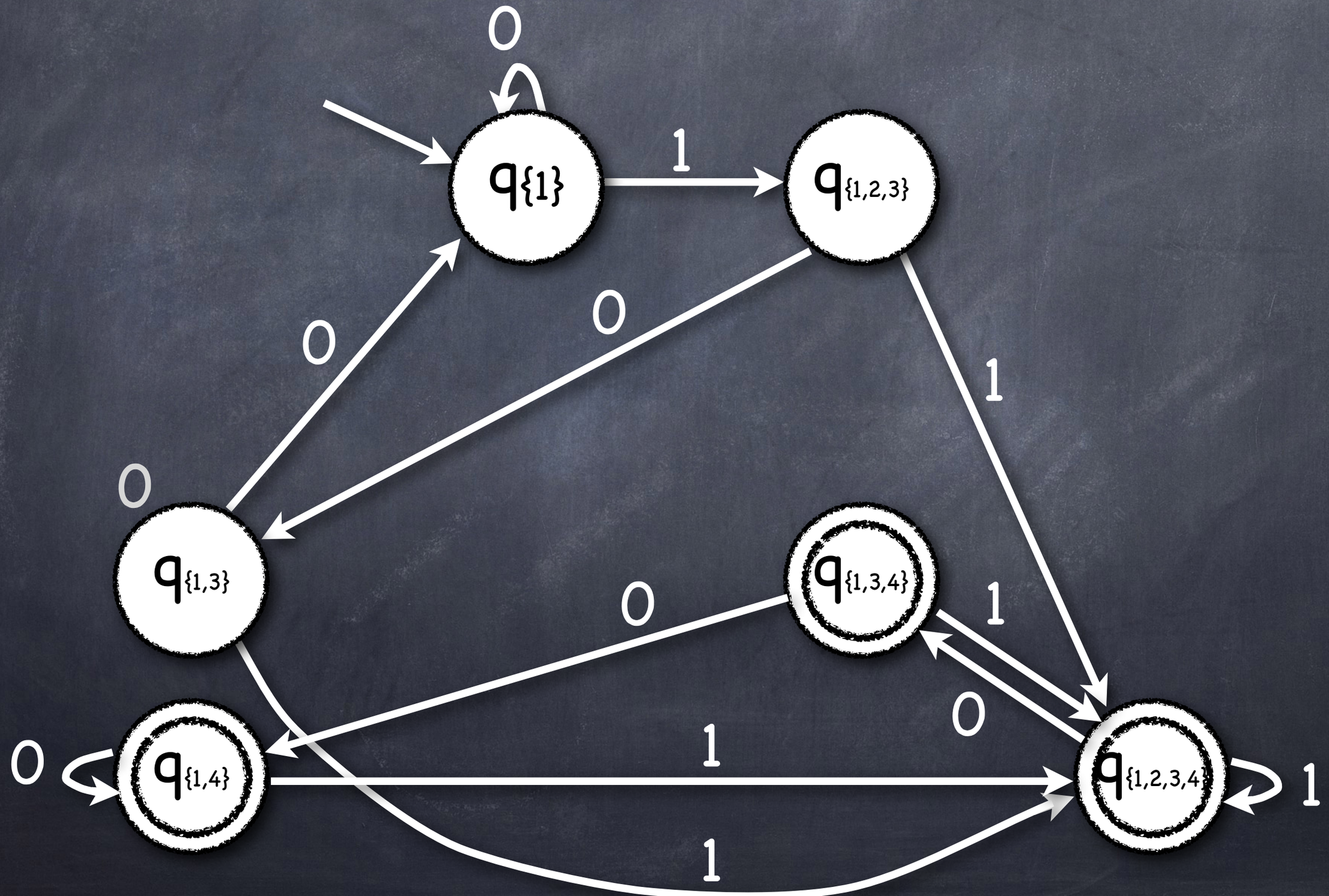
DFA minimization


0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

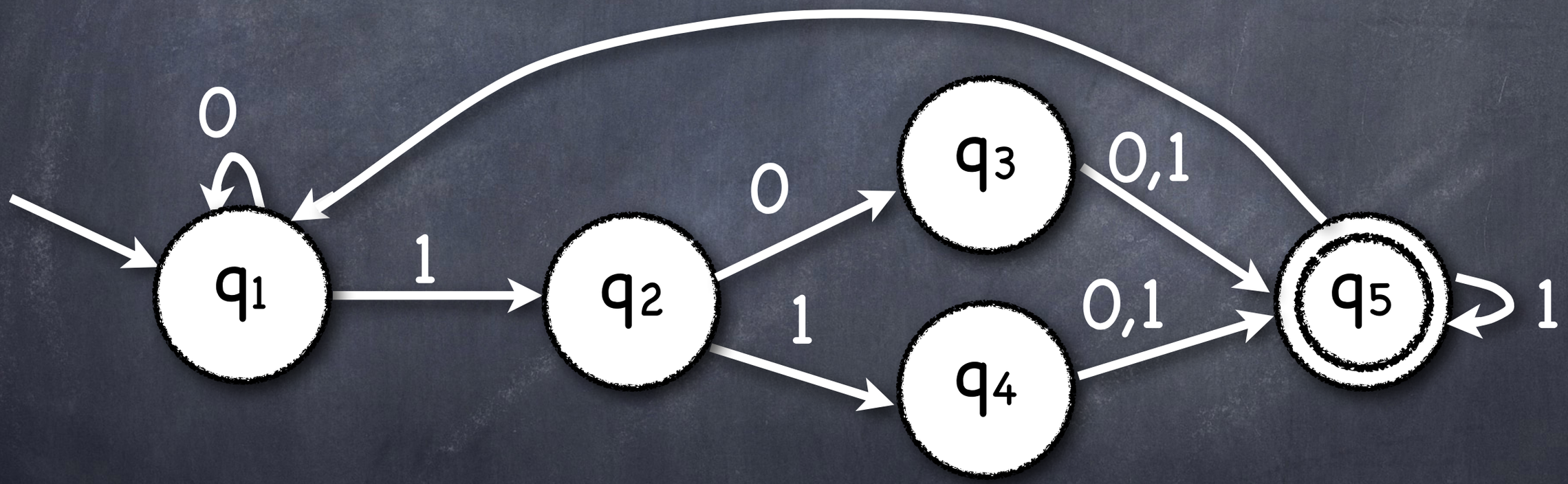

q{1}
q{1,2,3}


0
1
q{1}
q{1,2,3}
1

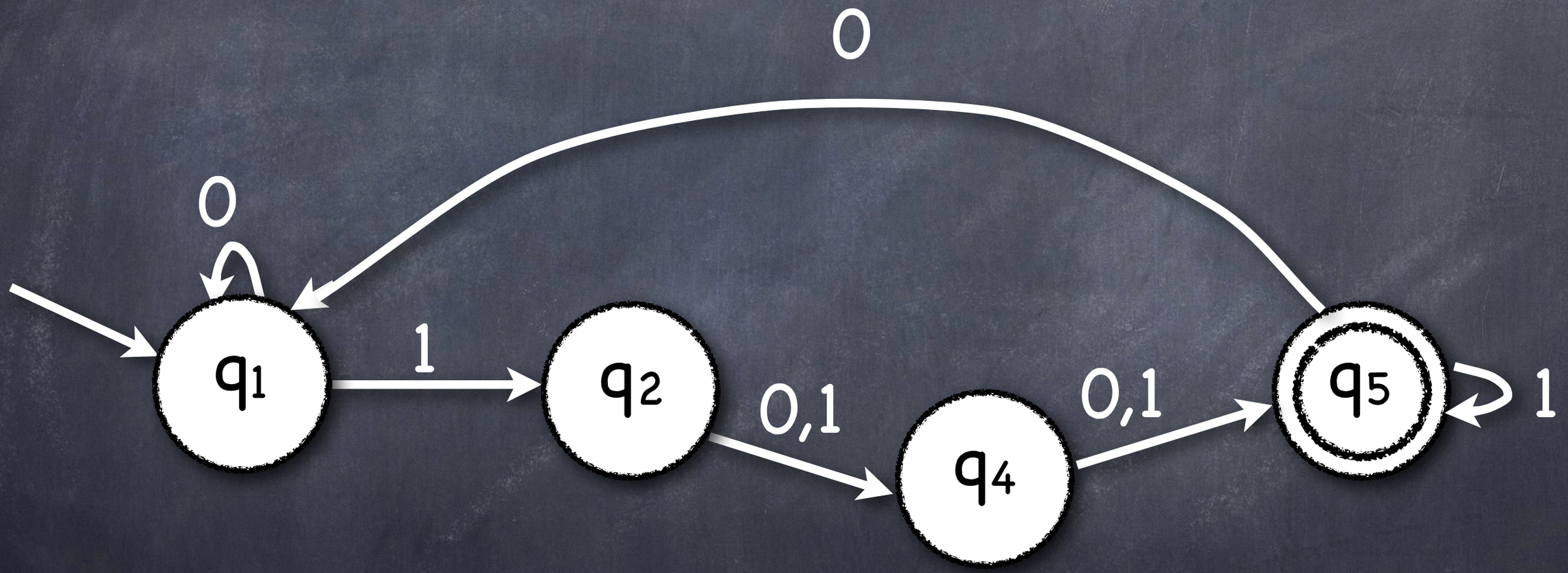
Reachable States



Redondant States



Redondant States



Myhill-Nerode Theorem

John R. Myhill

Anil Nerode

Myhill-Nerode Theorem

Myhill-Nerode Theorem

- Let x and y be strings and L be a language.

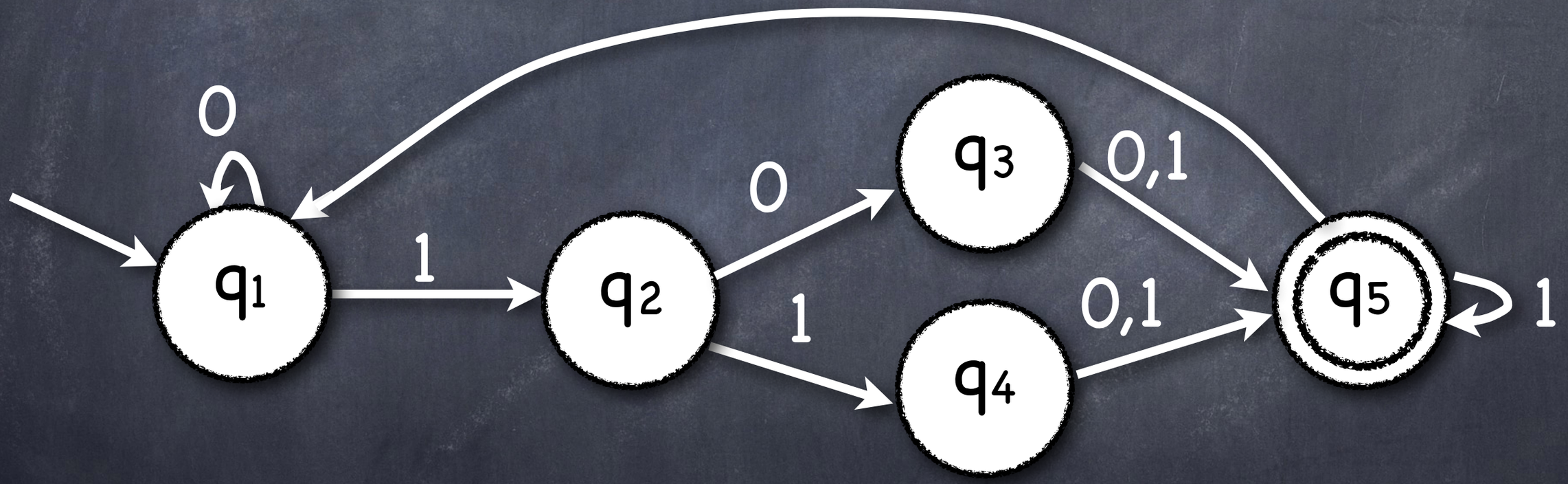
- Let x and y be strings and L be a language.

Myhill-Nerode Theorem

- Let x and y be strings and L be a language.
- We say that x and y are distinguishable by L if there exists a z such that $xz \in L$ and $yz \notin L$ or $yz \in L$ and $xz \notin L$.
- If x and y are indistinguishable by L we write $x \equiv_L y$, ($\equiv_L$ is an equivalence relation). If x, y are distinguishable by L we write $x \not\equiv_L y$.

Distinguishable Strings

$011 \not\equiv_L 1$

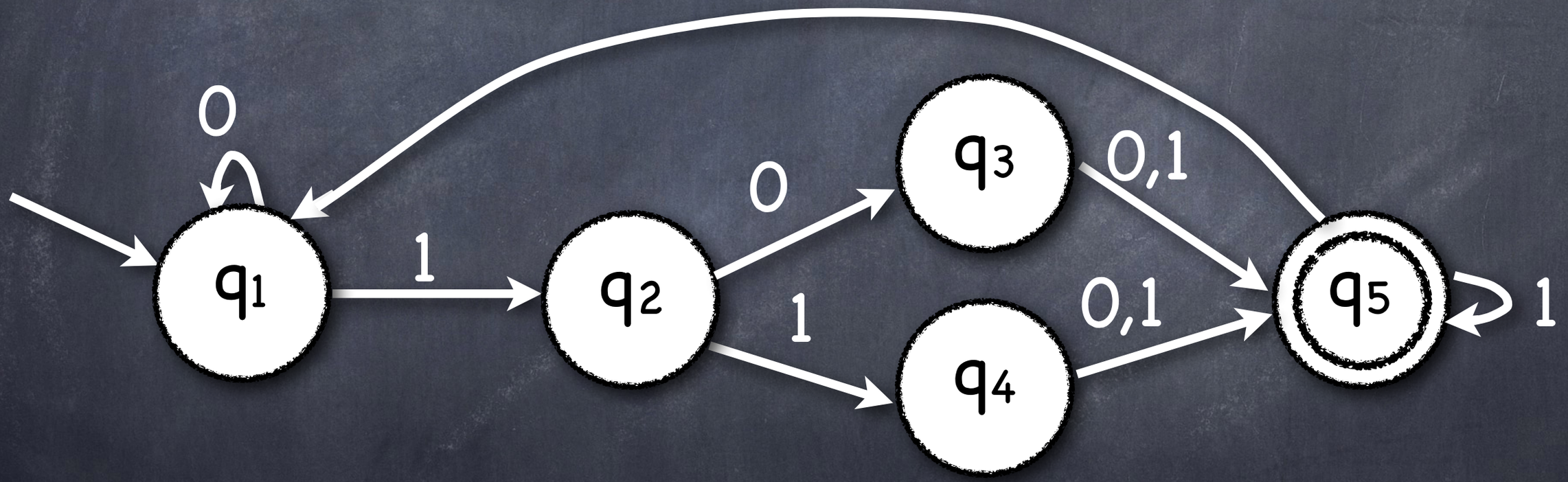


there exists a z such that $xz \in L$ and $yz \notin L$.

z=0 is such that $0110 \in L$ while $10 \notin L$.

Indistinguishable Strings

$011 \equiv_L 010$



There does not exist a z such that $xz \in L$ and $yz \notin L$ nor $yz \in L$ and $xz \notin L$. For all z, xz and yz are both in L or neither in L.

Myhill-Nerode Theorem

Myhill-Nerode Theorem

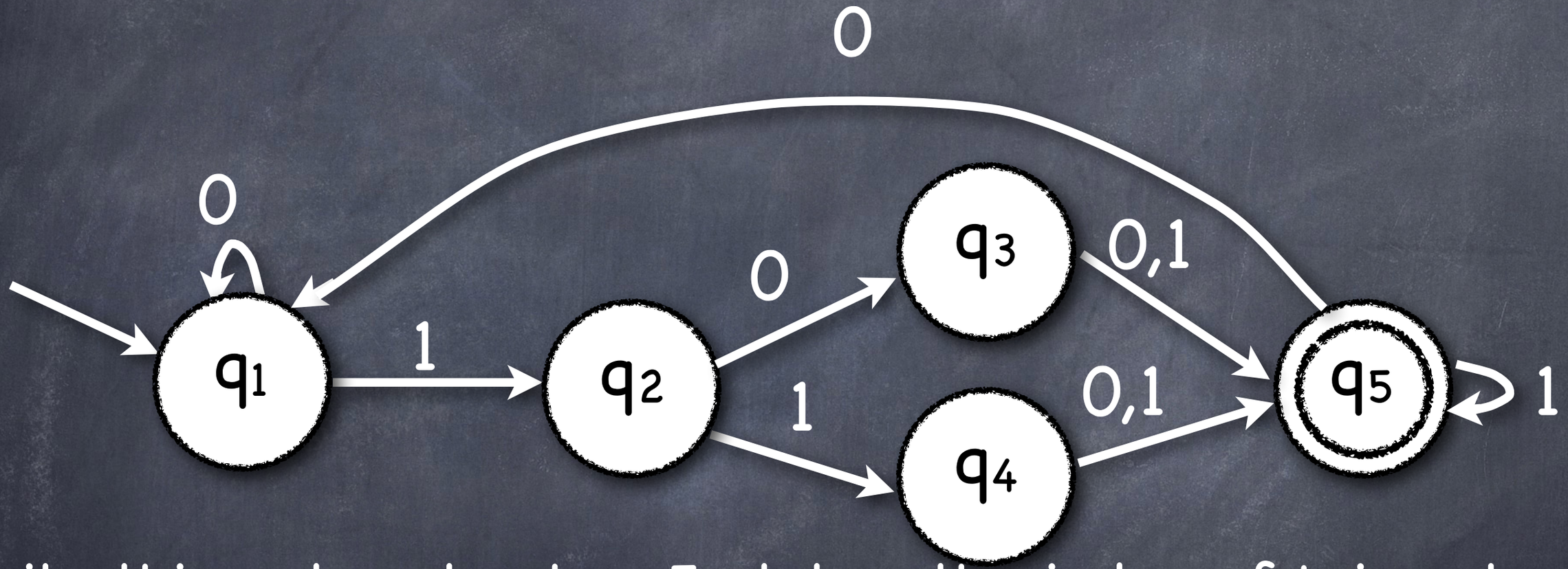
- Let L be a language and X a set of strings.

- Let L be a language and X a set of strings.

Myhill-Nerode Theorem

- Let L be a language and X a set of strings.
- We say that X is pairwise distinguishable by L if every two elements in X are distinguishable by L (For all x, x′ in X, $x \not\equiv_L x'$).
- Define the index of L to be the size of a maximum set X that is pairwise distinguishable by L. The index may be finite or infinite.

Distinguishable Strings



While this automaton has 5 states, the index of L is only 4:

ε, 1, 11 and 111

$111 \in L$ while $11 \notin L$, $111 \in L$ while $1 \notin L$, $111 \in L$ while $\varepsilon \notin L$,

$111 \in L$ while $11 \notin L$, $111 \in L$ while $1 \notin L$, $111 \in L$ while $11 \notin L$.

Myhill-Nerode Theorem

Myhill-Nerode Theorem

Myhill-Nerode Theorem

Myhill-Nerode Theorem

a. If L is recognized by a DFA with k states,
then L has index at most k.

b. If the index of L is a finite number k,
then it is recognized by a DFA with k states.

c. L is regular iff it has finite index.
This index is the size of the smallest DFA recognizing L.

Myhill-Nerode Theorem

a. If L is recognized by a DFA with k states,
then L has index at most k.

Myhill-Nerode Theorem

a. If L is recognized by a DFA with k states,

then L has index at most k.

- Let M be a k state DFA recognizing L.

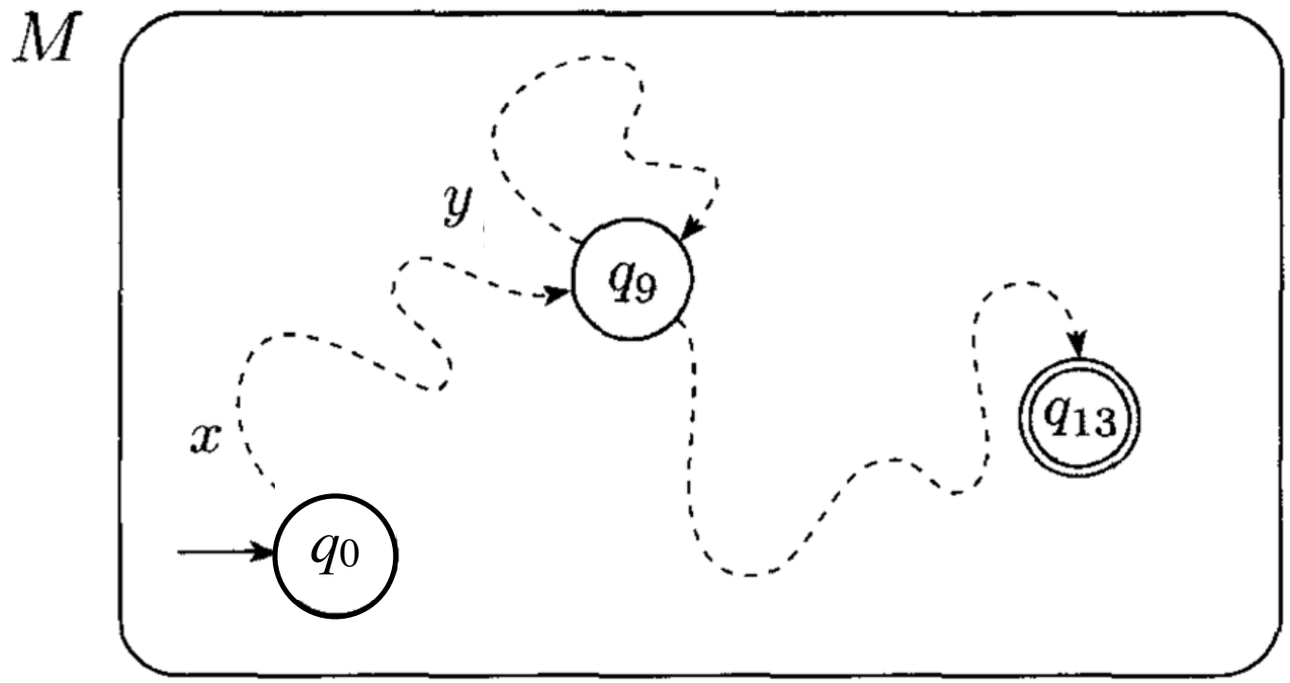
Myhill-Nerode Theorem

a. If L is recognized by a DFA with k states,

then L has index at most k.

- Let M be a k state DFA recognizing L.
- Suppose L has index larger than k.

My



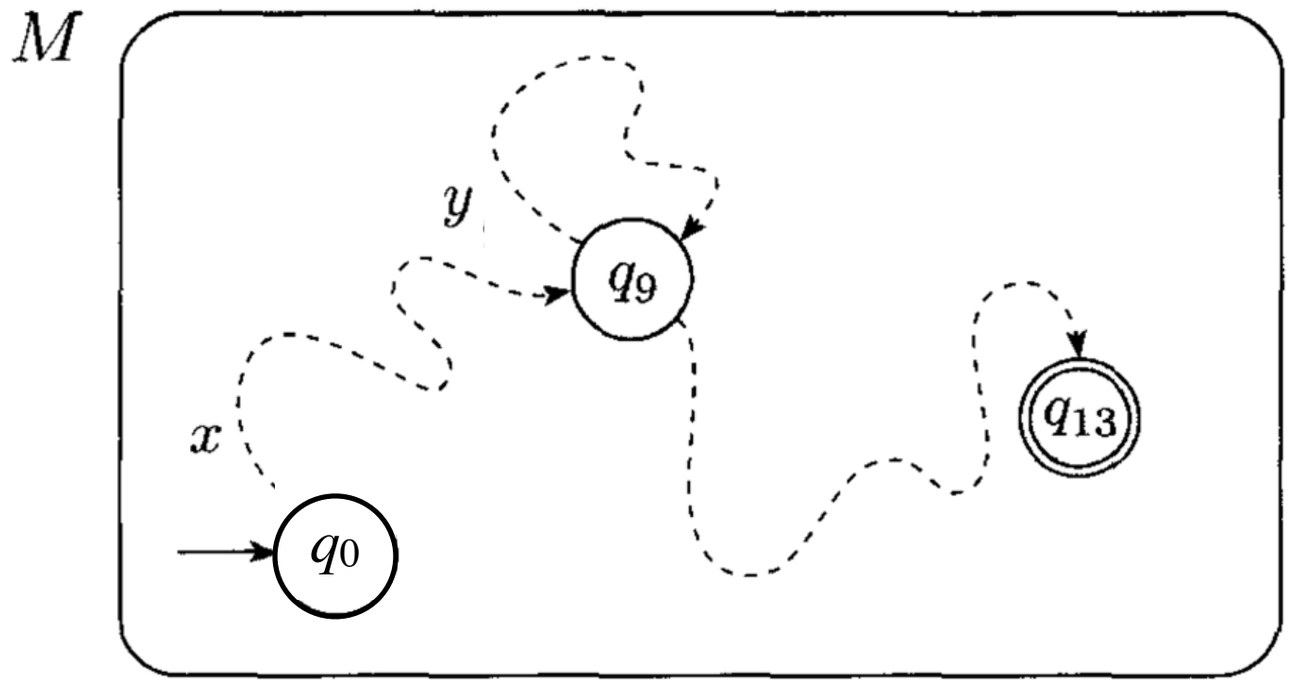
em

a. If L is r

then L has

- Let M be a k state DFA recognizing L.
- Suppose L has index larger than k.

My



em

a. If L is r

then L has

- Let M be a k state DFA recognizing L.
- Suppose L has index larger than k.
- Some X with k+1 elements is distinguishable by L. But since the number of states < k+1 there must exist x,y in X such that $\delta(q_0,x) = \delta(q_0,y)$. But then, x and y are not distinguishable.
  A contradiction.

Myhill-Nerode Theorem

b. If the index of L is a finite number k,

then it is recognized by a DFA with k states.

Myhill-Nerode Theorem

b. If the index of L is a finite number k,

then it is recognized by a DFA with k states.

- Let $X=\{s_1,\ldots,s_k\}$ be pairwise distinguishable by L.

Myhill-Nerode Theorem

b. If the index of L is a finite number k,

then it is recognized by a DFA with k states.

- Let $X=\{s_1,\ldots,s_k\}$ be pairwise distinguishable by L.

Myhill-Nerode Theorem

b. If the index of L is a finite number k,

then it is recognized by a DFA with k states.

- Let $X=\{s_1,\ldots,s_k\}$ be pairwise distinguishable by L.

Myhill-Nerode Theorem

b. If the index of L is a finite number k,

then it is recognized by a DFA with k states.

- Let $X=\{s_1,\ldots,s_k\}$ be pairwise distinguishable by L.
- Let $Q=\{q_1,\ldots,q_k\}$ be the states of a DFA recognizing L and define $\delta(q_i,a)=q_j$ s.t. $s_j \equiv_L s_i a$.
- Let q_0 be the q_i s.t. $s_i \equiv_L \varepsilon$. Let $F=\{ q_i \mid s_i \in L \}$.
- M is s.t. $\{ s \mid \delta(q_0,s)=q_i \} = \{ s \mid s \equiv_L s_i \}$.

Myhill-Nerode Theorem

c. L is regular iff it has finite index.
This index is the size of the smallest DFA recognizing L.

($\Longrightarrow$) L is regular implies the existence of a DFA recognizing L. By (a), L has index at most k.
($\Longleftarrow$) If L has index k then by (b) there exists a DFA with k states (i.e. L is regular).

- As for the minimality, if the index of L is not the size of the minimal DFA then there exists a DFA with index-1 states recognizing L. But this is impossible by part (a).

Minimizing via Myhill-Nerode Theorem

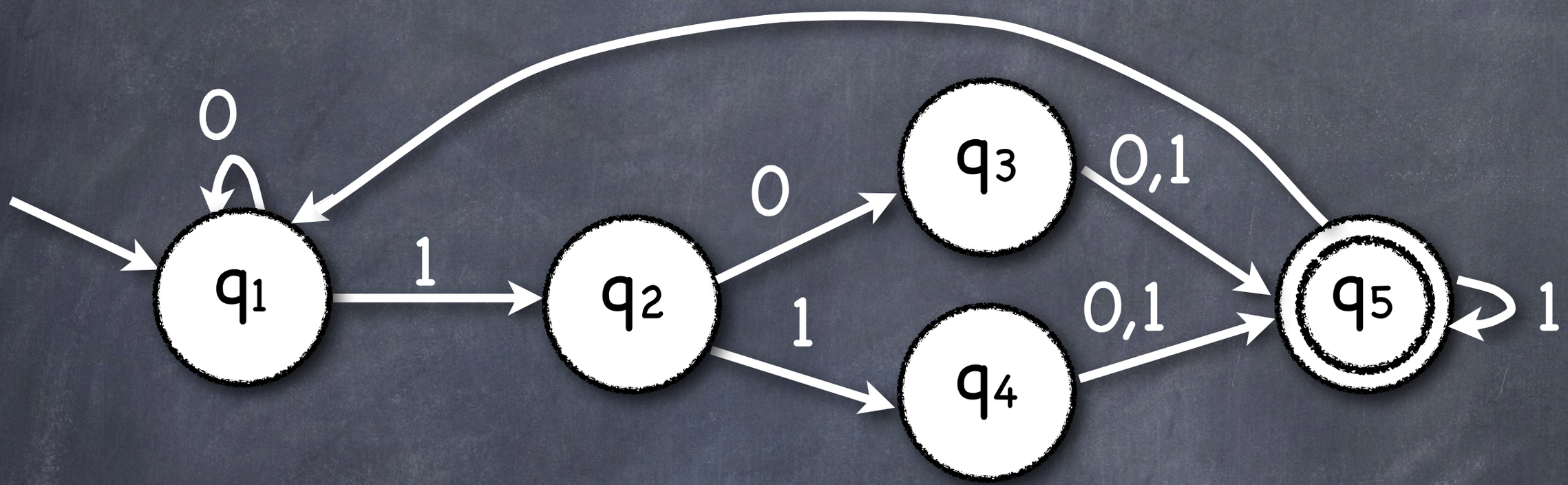
Minimizing via Myhill-Nerode Theorem

Minimizing via Myhill-Nerode Theorem

- Let L be a regular language. Compute the index of L by finding the set X of all the strings that are pairwise distinguishable by L.

- All strings considered as x, y, xz and yz may be shorter than the number of states of a DFA accepting L. Every string which is longer is equivalent to a shorter one obtained by pumping down.

Computing Index



If we consider all $63=2^6-1$ strings of length up to 5, we get:

$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

Minimizing via Myhill-Nerode Theorem

- Let L be a regular language. Compute the index of L by finding the set X of all the strings that are pairwise distinguishable by L.
- Using part (b) of the Myhill-Nerode Theorem we construct a minimal DFA to accept L.

Minimal DFA

Minimal DFA

(b) Let X={ε,1,10,100} be pairwise distinguishable by L.

Minimal DFA

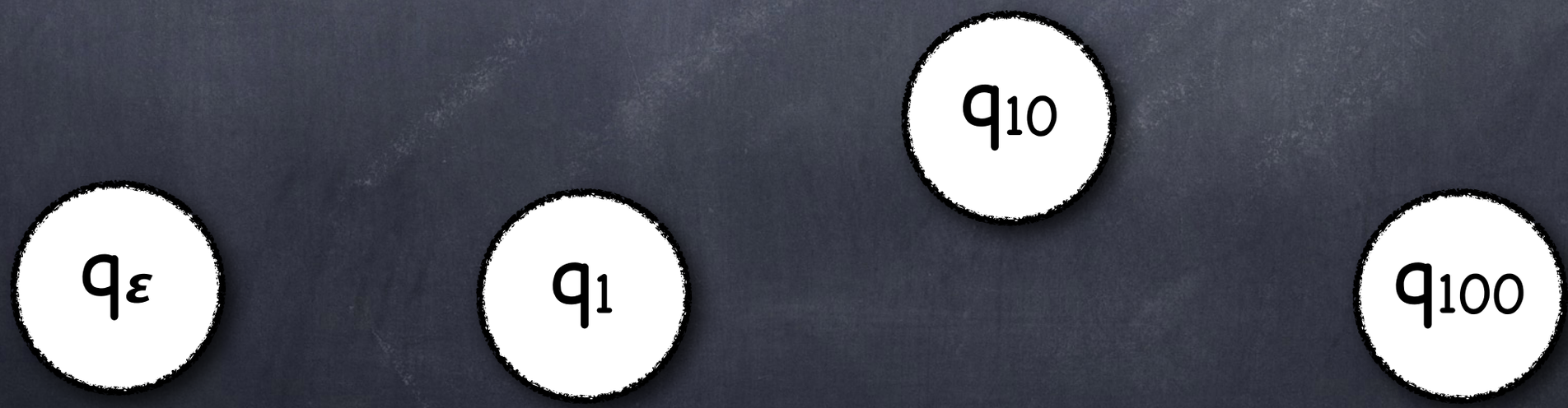
(b) Let $X=\{\varepsilon,1,10,100\}$ be pairwise distinguishable by L.

- Let $Q=\{q_\varepsilon,q_1,q_{10},q_{100}\}$ be the states of a DFA recognizing L

Minimal DFA

(b) Let $X=\{\varepsilon,1,10,100\}$ be pairwise distinguishable by L.

- Let $Q=\{q_\varepsilon,q_1,q_{10},q_{100}\}$ be the states of a DFA recognizing L



Minimal DFA

(b) Let X={ε,1,10,100} be pairwise distinguishable by L.

- Let Q={q_ε,q_1,q_{10},q_{100}} be the states of a DFA recognizing L
- Let q_ε be the initial state and F={ q_{100} }.

Minimal DFA

(b) Let $X=\{\varepsilon,1,10,100\}$ be pairwise distinguishable by L.

- Let $Q=\{q_\varepsilon,q_1,q_{10},q_{100}\}$ be the states of a DFA recognizing L
- Let q_ε be the initial state and $F=\{ q_{100} \}$.

Minimal DFA

(b) Let $X=\{\varepsilon,1,10,100\}$ be pairwise distinguishable by L.

- Let $Q=\{q_\varepsilon,q_1,q_{10},q_{100}\}$ be the states of a DFA recognizing L
- Let q_ε be the initial state and $F=\{ q_{100} \}$.
- Define $\delta(q_w,a)=q_{w'}$ s.t. $w' \equiv_L wa$.

$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.

$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.

$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.

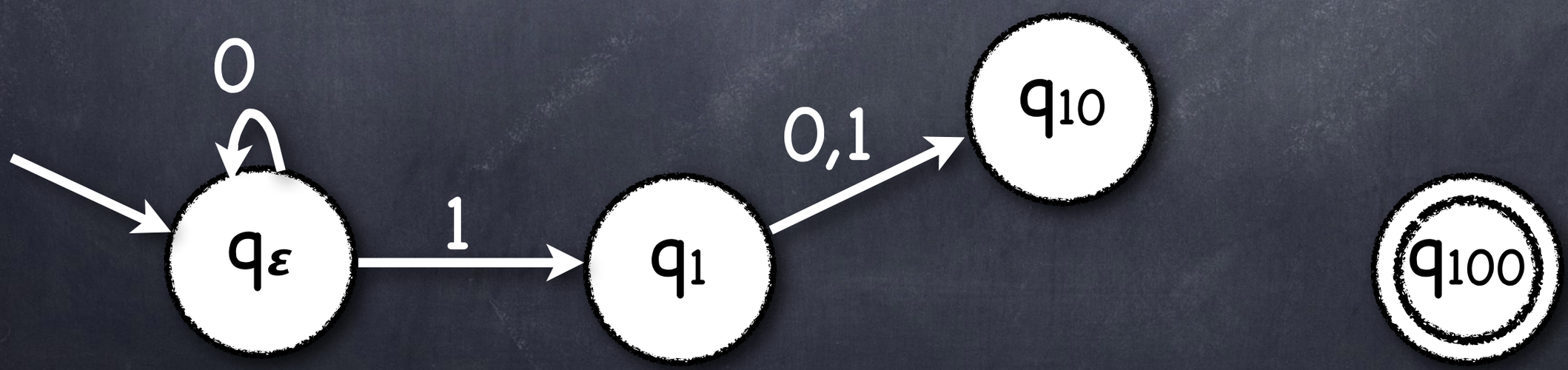
$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.



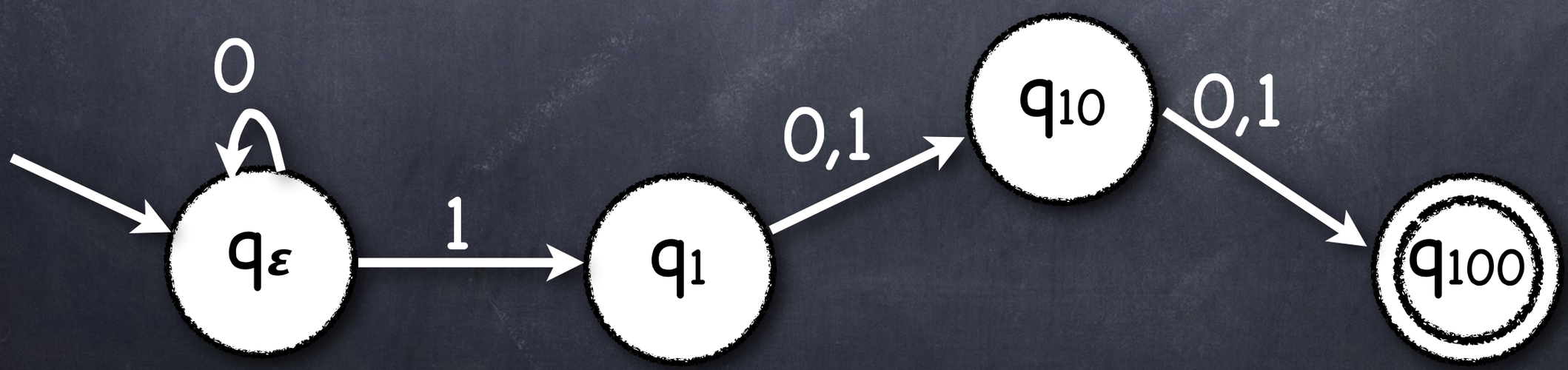
$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.



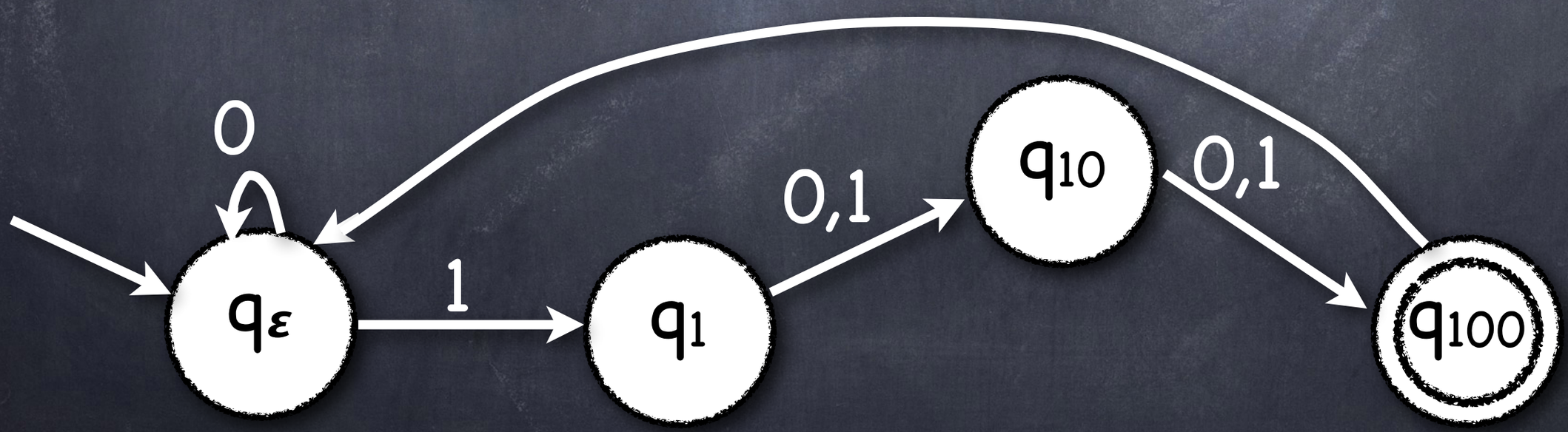
$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.



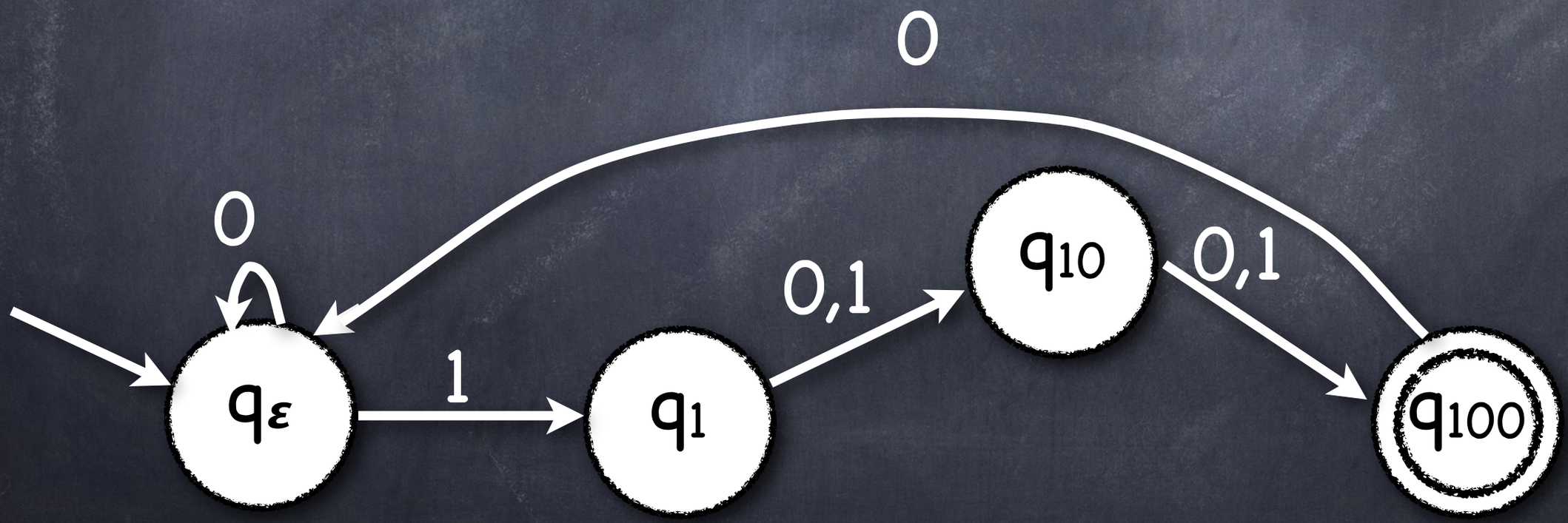
$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.



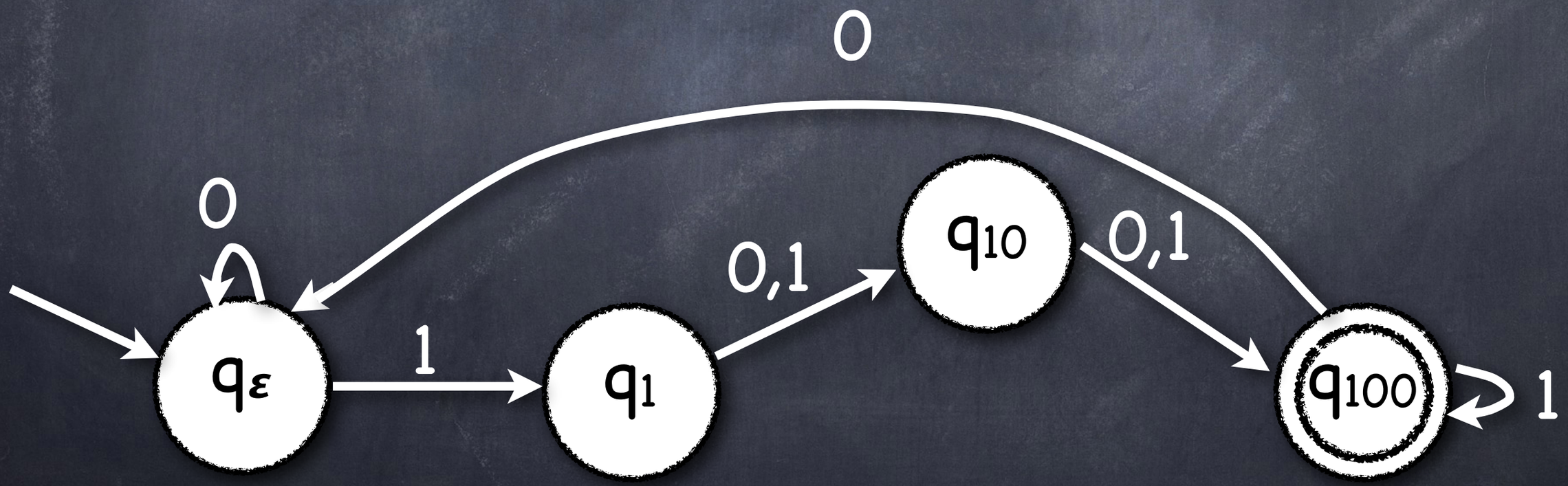
$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.



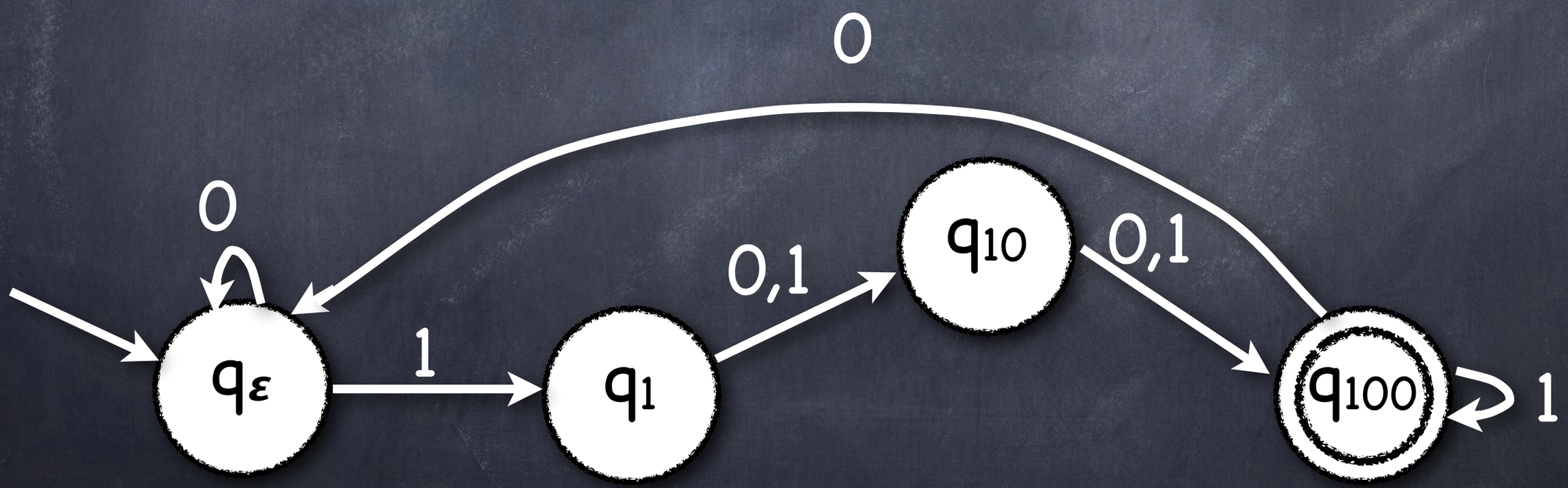
$\varepsilon \equiv_L 0 \equiv_L 00 \equiv_L 000 \equiv_L 0000 \equiv_L 1000 \equiv_L 1010 \equiv_L 1100 \equiv_L 1110 \equiv_L$
$00000 \equiv_L 01000 \equiv_L 01010 \equiv_L 01100 \equiv_L 01110 \equiv_L 10000 \equiv_L 10010 \equiv_L 10100 \equiv_L 10110 \equiv_L 11000 \equiv_L 11010 \equiv_L 11100 \equiv_L 11110$

$1 \equiv_L 01 \equiv_L 001 \equiv_L 0001 \equiv_L 00001 \equiv_L 10001 \equiv_L 10101 \equiv_L 11001 \equiv_L 11101$

$10 \equiv_L 11 \equiv_L 010 \equiv_L 011 \equiv_L 0010 \equiv_L 0011 \equiv_L 00010 \equiv_L 00011$

$100 \equiv_L 101 \equiv_L 110 \equiv_L 111 \equiv_L 0100 \equiv_L 0101 \equiv_L 0110 \equiv_L 0111 \equiv_L 1001 \equiv_L 1011 \equiv_L 1101 \equiv_L 1111 \equiv_L$
$00100 \equiv_L 00101 \equiv_L 00110 \equiv_L 00111 \equiv_L 01001 \equiv_L 01011 \equiv_L 01101 \equiv_L 01111 \equiv_L 10011 \equiv_L 10111 \equiv_L 11011 \equiv_L 11111$

- Define $\delta(q_w, a) = q_{w'}$ s.t. $w' \equiv_L wa$.
- M is s.t. $\{ s \mid \delta(q_\varepsilon, s) = q_w \} = \{ s \mid s \equiv_L w \}$.



Application of the Myhill-Nerode Theorem

$B = \{ 0^n1^n \mid n \geq 0 \}$ is non-regular because it has infinite index.

Consider the set $X=\{ 0^n \mid n \geq 0 \}$. It's an infinite set that is pairwise distinguishable by B.

Proof: For all n, 0^n is distinguishable from all previous 0^i, $0 \leq i \leq n-1$, because there exists a $z=1^n$ such that $0^n z \in B$ while $0^i z \notin B$, $0 \leq i \leq n-1$.

QED

Application of the Myhill-Nerode Theorem

$F = \{ ww \mid w \in \Sigma^* \}$ is non-regular because it has infinite index.

Consider the set $X=\{ 0^n1 \mid n \geq 0 \}$. It's an infinite set that is pairwise distinguishable by F.

Proof: For all n, 0^n1 is distinguishable from all previous 0^i1, $0 \leq i \leq n-1$, because there exists a $z=0^n1$ such that $0^n1z \in B$ while $0^i1z \notin B$, $0 \leq i \leq n-1$.

QED

Regular Expressions

Regular Expressions

DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

Regular Expressions

DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

$R_1^+ = (R_1 \circ R_1^*)$

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

EXAMPLE 1.53

In the following instances we assume that the alphabet Σ is $\{0,1\}$.

1. $0^*10^* = \{w | \ w \text{ contains a single } 1\}$.
2. $\Sigma^*1\Sigma^* = \{w | \ w \text{ has at least one } 1\}$.
3. $\Sigma^*001\Sigma^* = \{w | \ w \text{ contains the string } 001 \text{ as a substring}\}$.
4. $1^*(01^+)^* = \{w | \ \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.
5. $(\Sigma\Sigma)^* = \{w | \ w \text{ is a string of even length}\}$.[5]
6. $(\Sigma\Sigma\Sigma)^* = \{w | \ \text{the length of } w \text{ is a multiple of three}\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w | \ w \text{ starts and ends with the same symbol}\}$.
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$.
   The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^*.
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.
    Concatenating the empty set to any set yields the empty set.
12. $\emptyset^* = \{\varepsilon\}$.
    The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.

Regular Expressions vs Regular Languages

THEOREM 1.54

A language is regular if and only if some regular expression describes it.

LEMMA 1.55

If a language is described by a regular expression, then it is regular.

LEMMA 1.60

If a language is regular, then it is described by a regular expression.

Regular Expressions vs Regular Languages

LEMMA 1.55

If a language is described by a regular expression, then it is regular.

LEMMA 1.60

If a language is regular, then it is described by a regular expression.

Regular Expressions vs Regular Languages

LEMMA 1.55

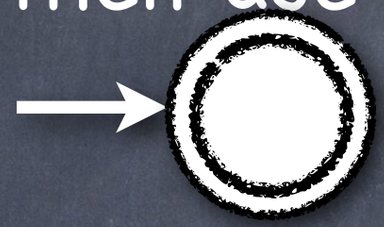
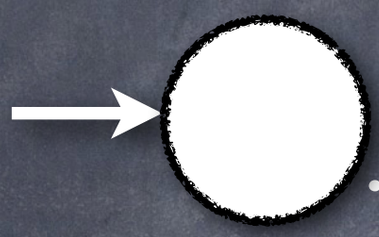
If a language is described by a regular expression, then it is regular.

LEMMA 1.60

If a language is regular, then it is described by a regular expression.

Regular Expressions generate Reg. Languages

- Let R_A generating L_A.
- If R_A is a symbol "a" then use 

.
- If R_A is "ε" then use  .
- If R_A is "$\varnothing$" then use  .
- If R_A is $R_1 \cup R_2$ then use Thm 1.45 and recursively use N_1 and N_2 s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$.
- If R_A is $R_1 \circ R_2$ then use Thm 1.47 and recursively use N_1 and N_2 s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$.
- If R_A is R^* then use Thm 1.49 and recursively use N s.t. $L(N)=L(R)$.

Automata recognize Regular Expressions

LEMMA 1.60

If a language is regular, then it is described by a regular expression.

Generalized NFA

Example of GNFA

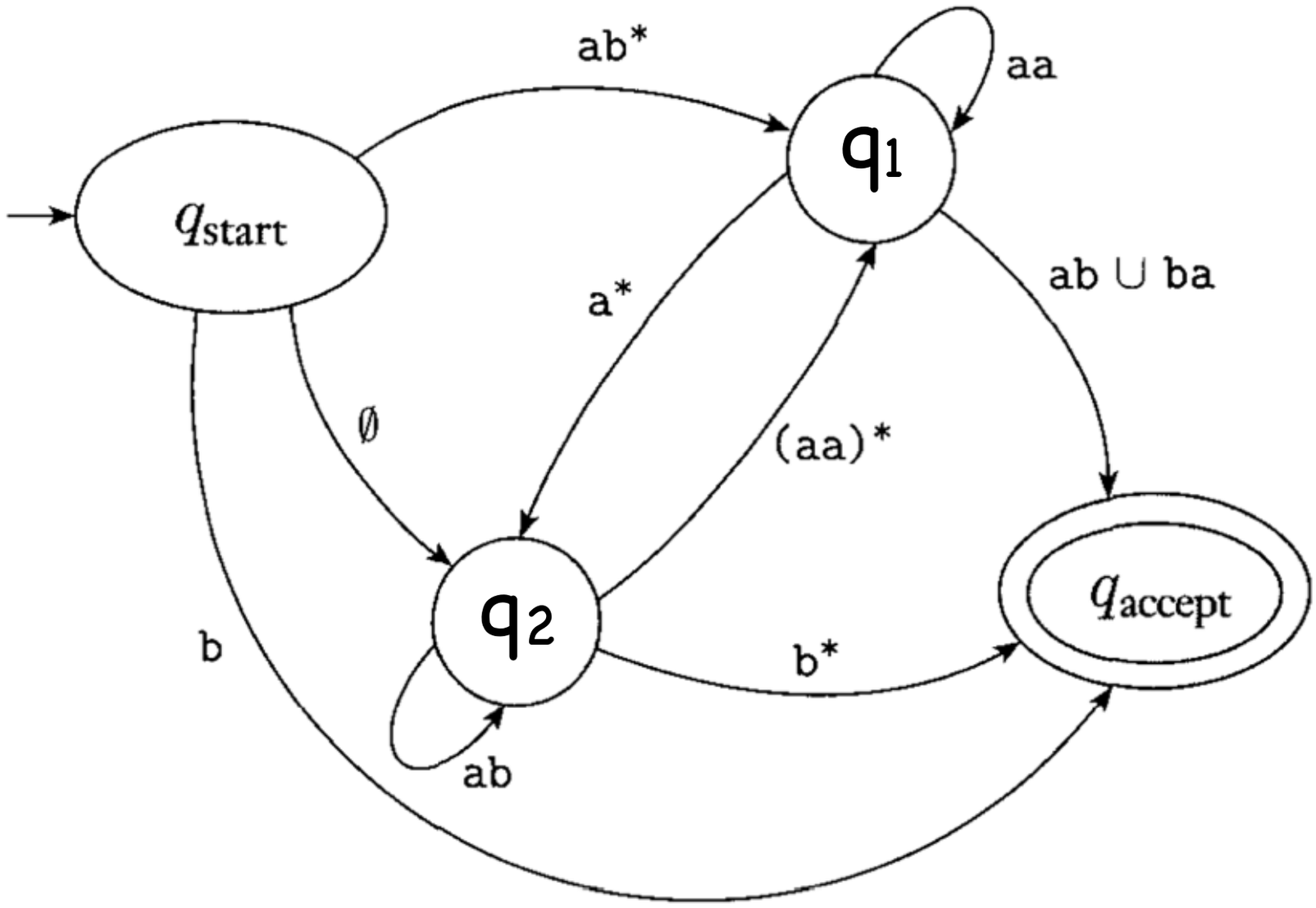
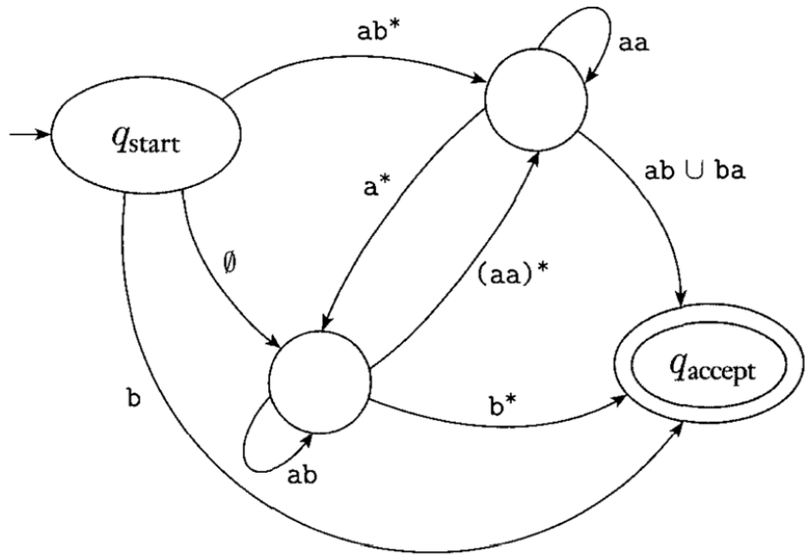


FIGURE 1.61
A generalized nondeterministic finite automaton

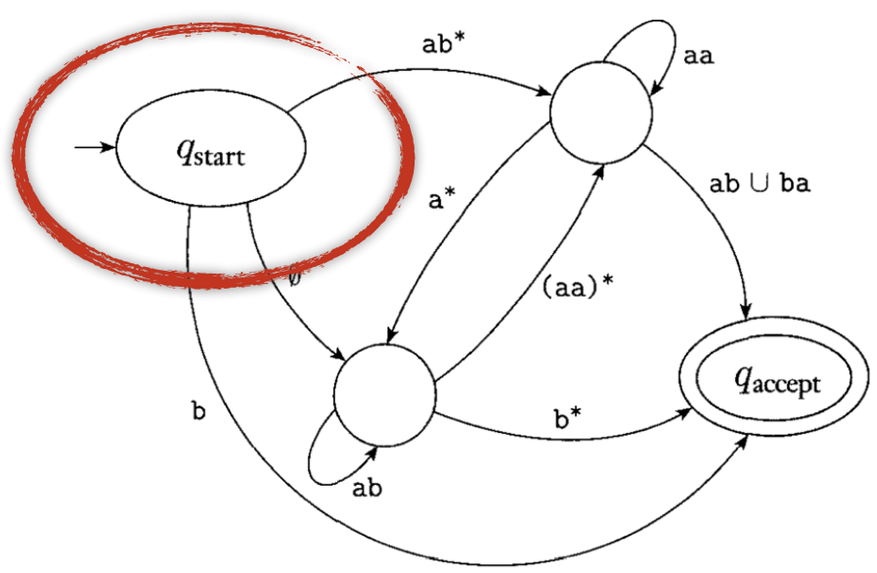
Generalized NFA



For convenience we require that GNFAs always have a special form that meets the following conditions.

- The start state has transition arrows going to every other state but no arrows coming in from any other state.
- There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
- Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

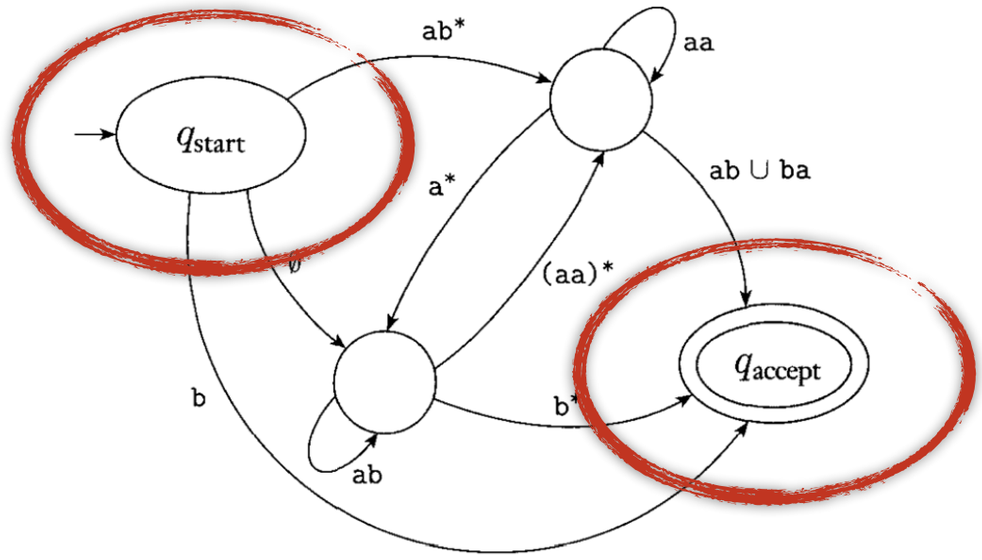
Generalized NFA



For convenience we require that GNFAs always have a special form that meets the following conditions.

- The start state has transition arrows going to every other state but no arrows coming in from any other state.
- There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
- Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

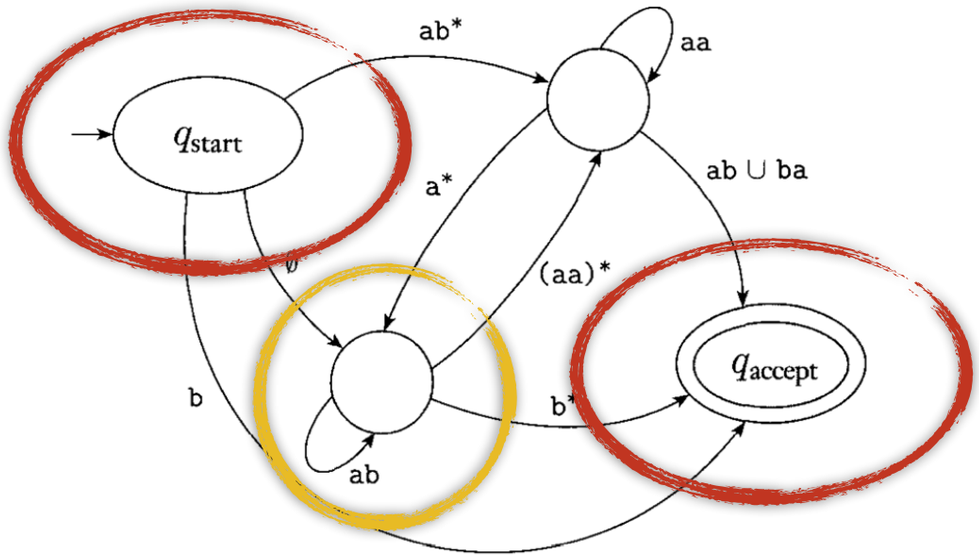
Generalized NFA



For convenience we require that GNFAs always have a special form that meets the following conditions.

- The start state has transition arrows going to every other state but no arrows coming in from any other state.
- There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
- Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

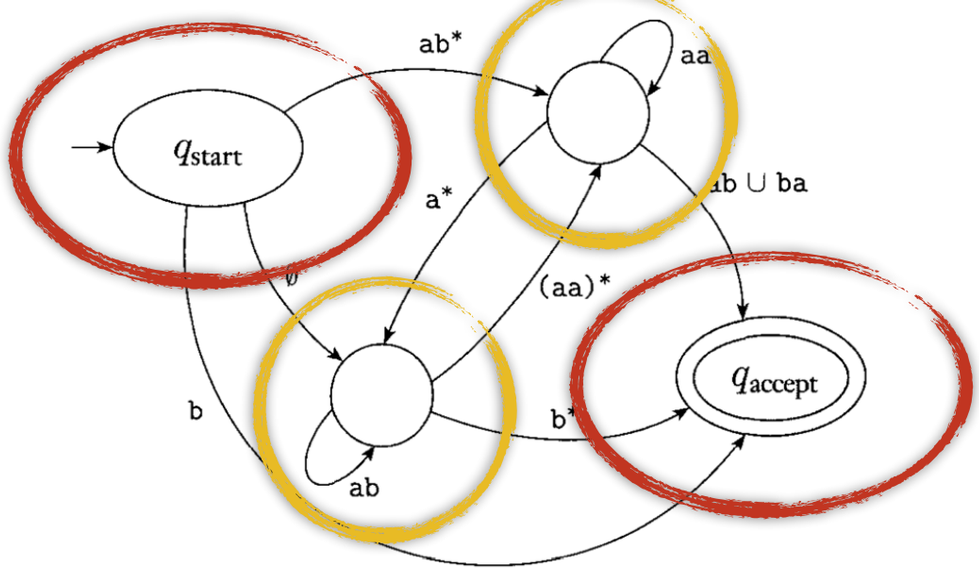
Generalized NFA



For convenience we require that GNFAs always have a special form that meets the following conditions.

- The start state has transition arrows going to every other state but no arrows coming in from any other state.
- There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
- Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

Generalized NFA



For convenience we require that GNFAs always have a special form that meets the following conditions.

- The start state has transition arrows going to every other state but no arrows coming in from any other state.
- There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
- Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

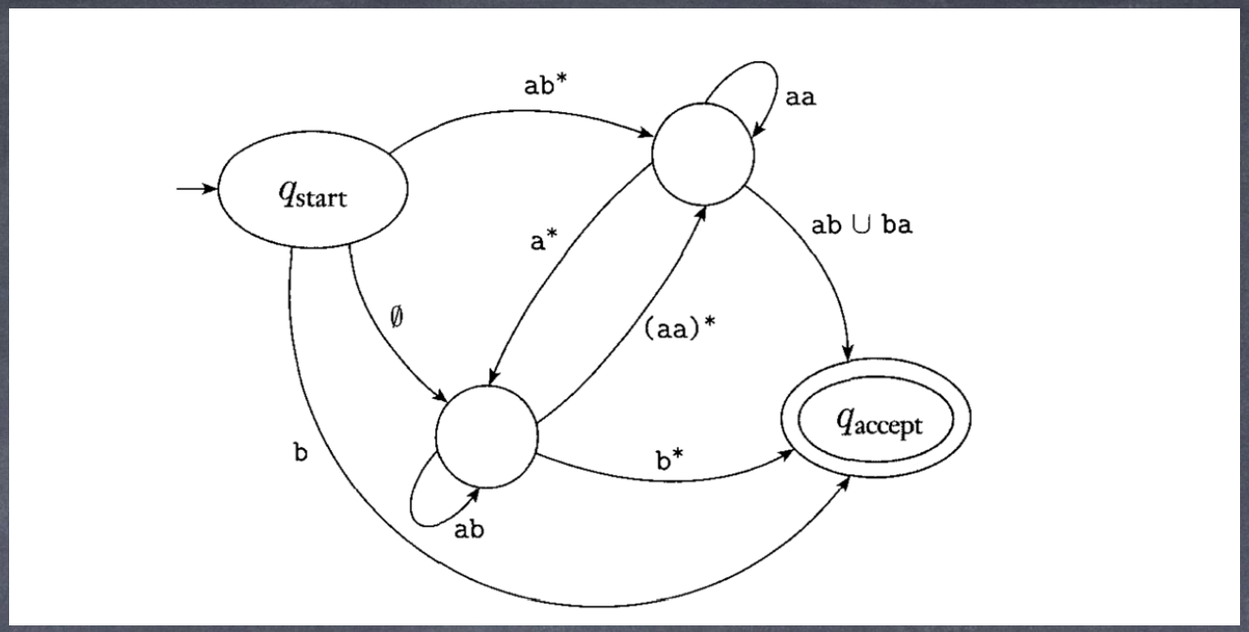
Generalized NFA

DEFINITION 1.64

A ***generalized nondeterministic finite automaton*** is a 5-tuple, $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where

1. Q is the finite set of states,
2. Σ is the input alphabet,
3. $\delta\colon (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathcal{R}$ is the transition function,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

Definition of GNFA

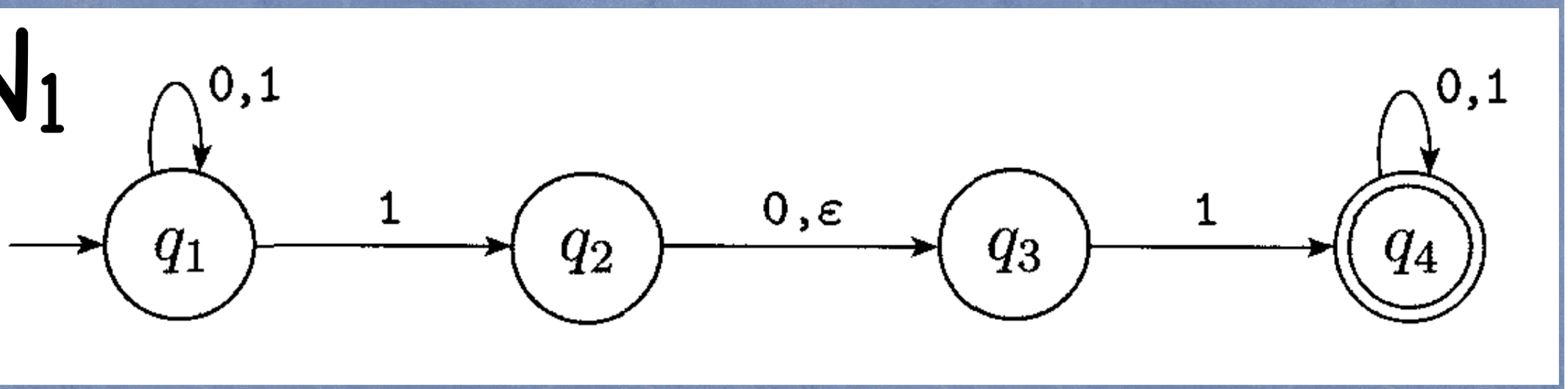


- Let $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ be a generalized nondeterministic finite state automaton and let $w = w_1 w_2 \ldots w_n$ $(n \geq 0)$ be a string where each sub-string $w_i \in \Sigma^*$.

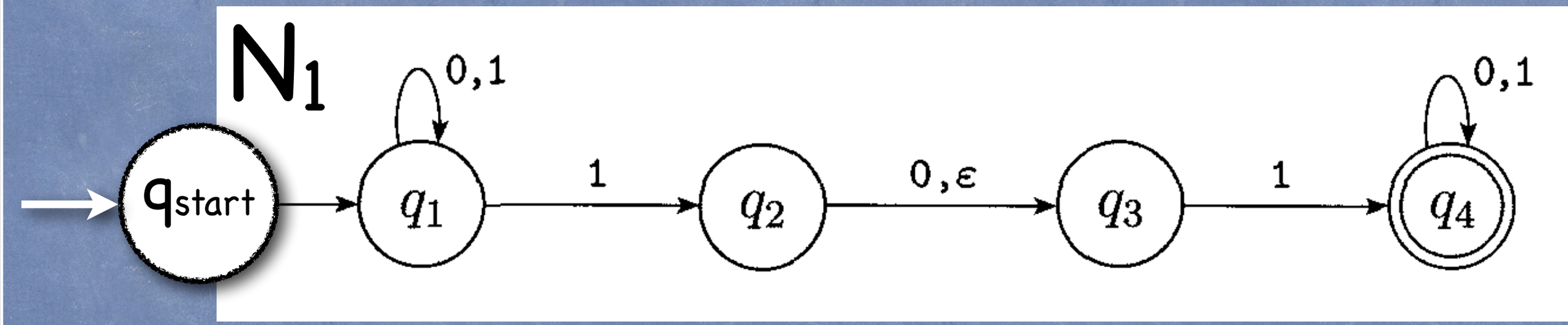
- G **accepts** w if $\exists\ s_0, s_1, \ldots, s_n$ s.t.
  1. $s_0 = q_{start}$
  2. $w_i \in L(\delta(s_{i-1}, s_i))$ for $i = 1 \ldots n$, and
  3. $s_n = q_{accept}$

Example NFA→GNFA

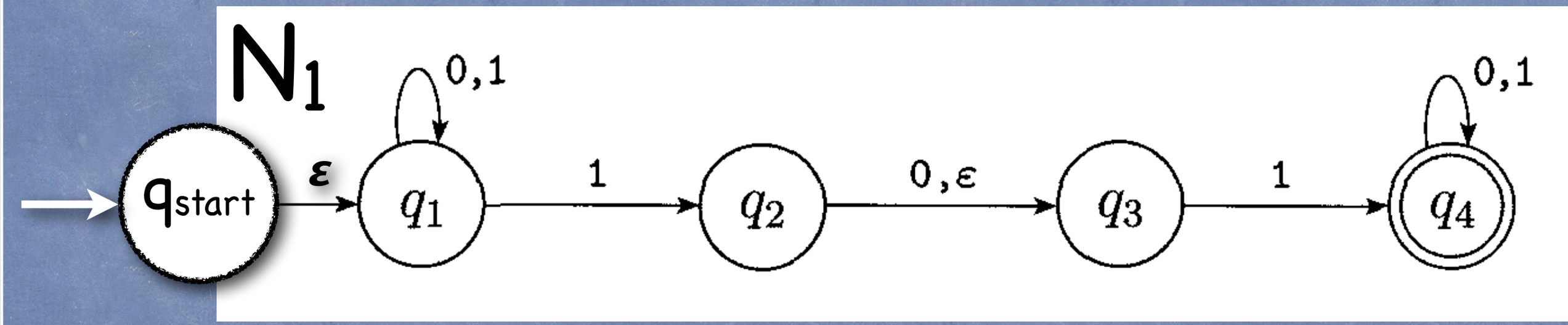
N_1



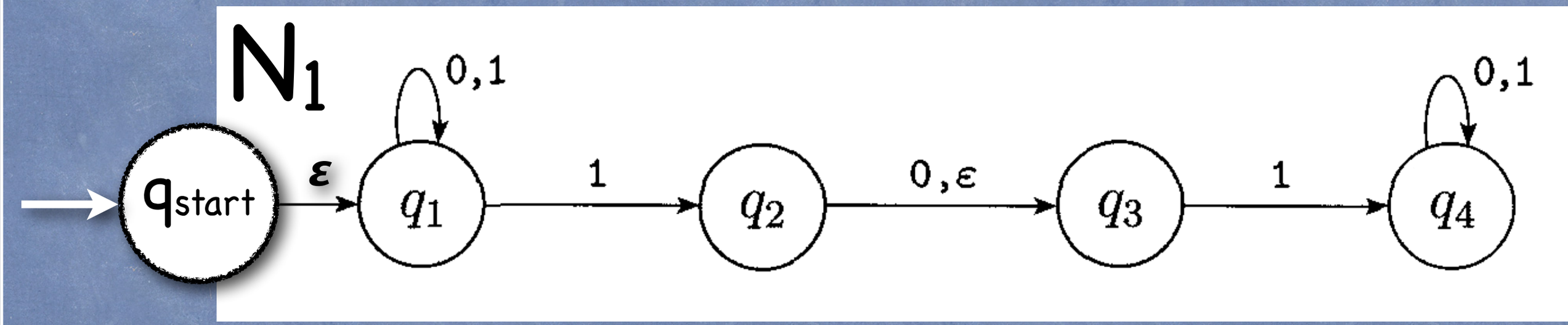
Example NFA→GNFA



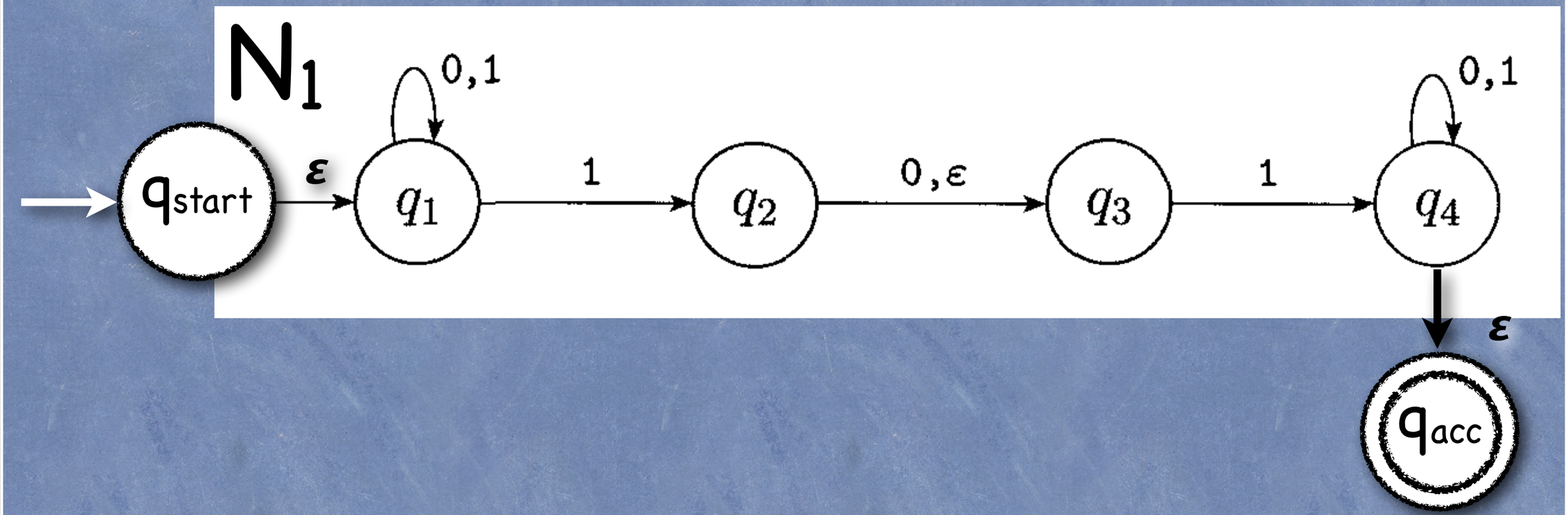
Example NFA→GNFA



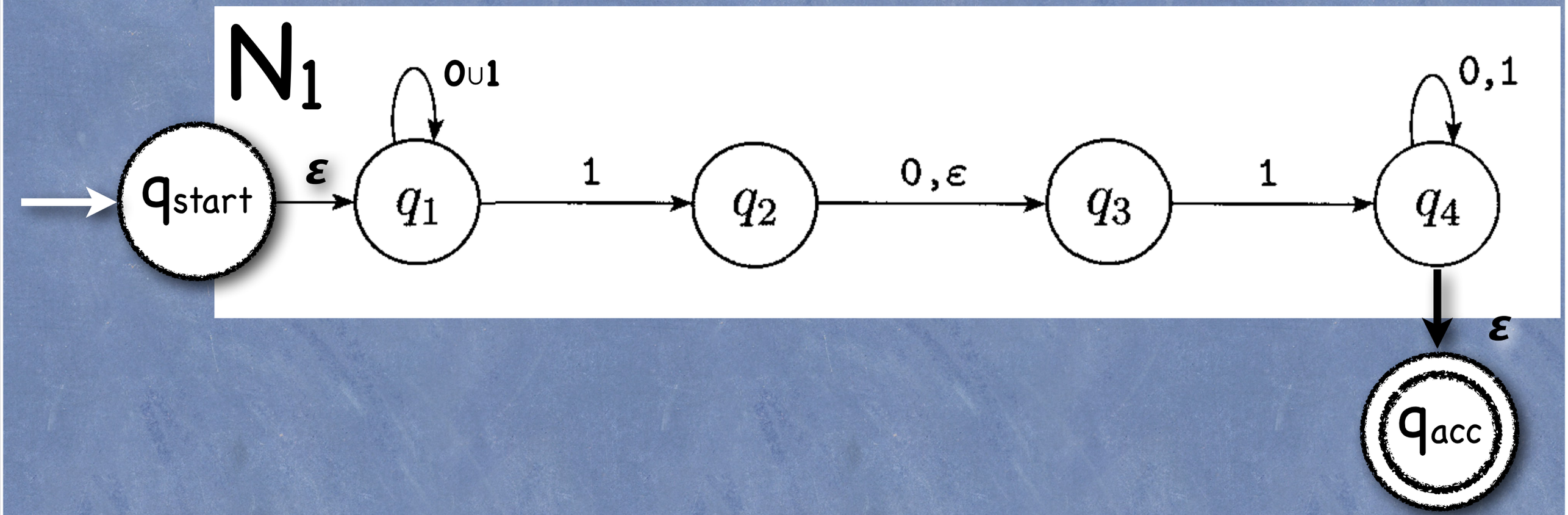
Example NFA→GNFA



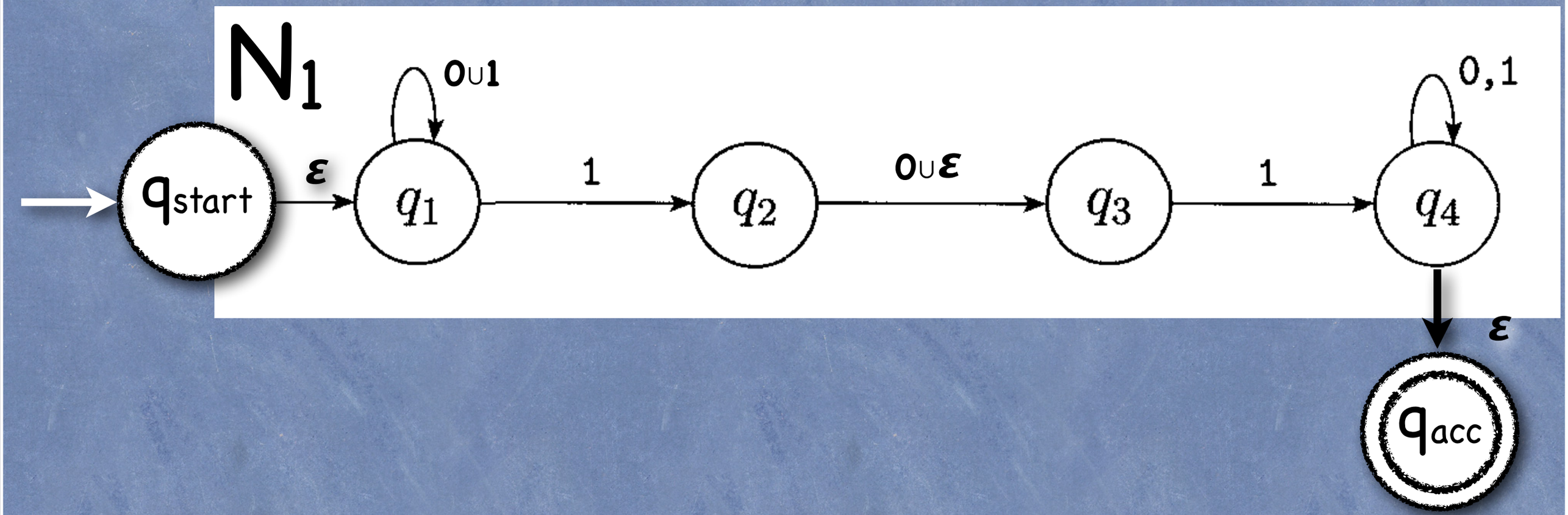
Example NFA→GNFA



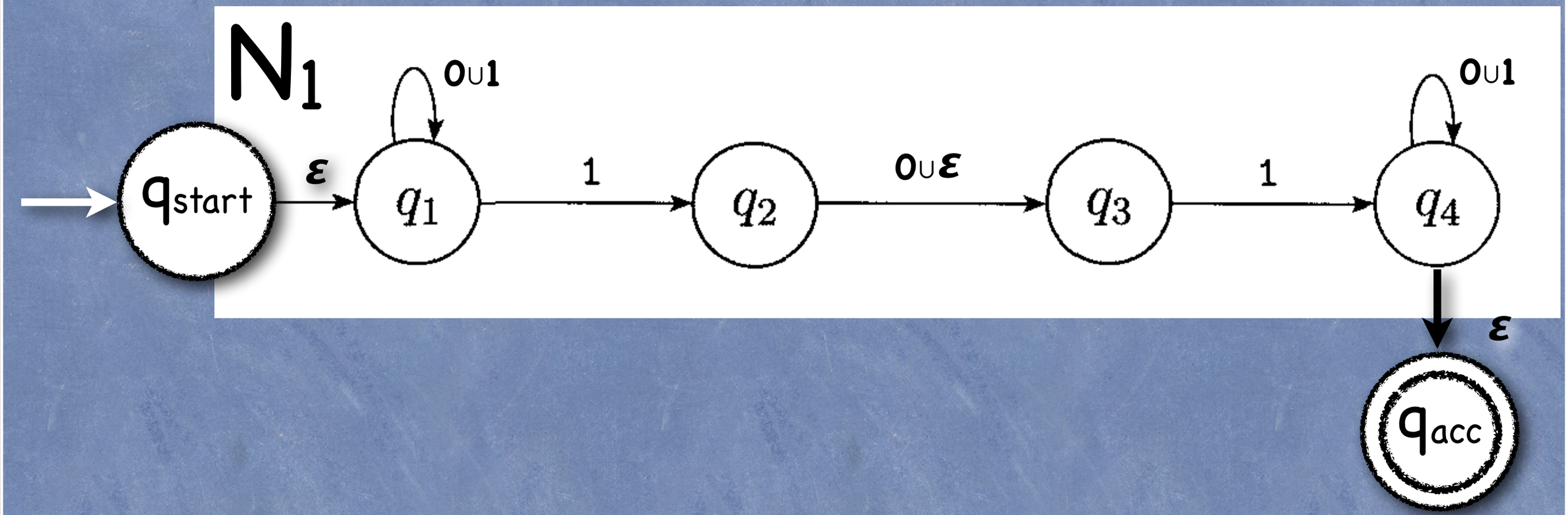
Example NFA→GNFA



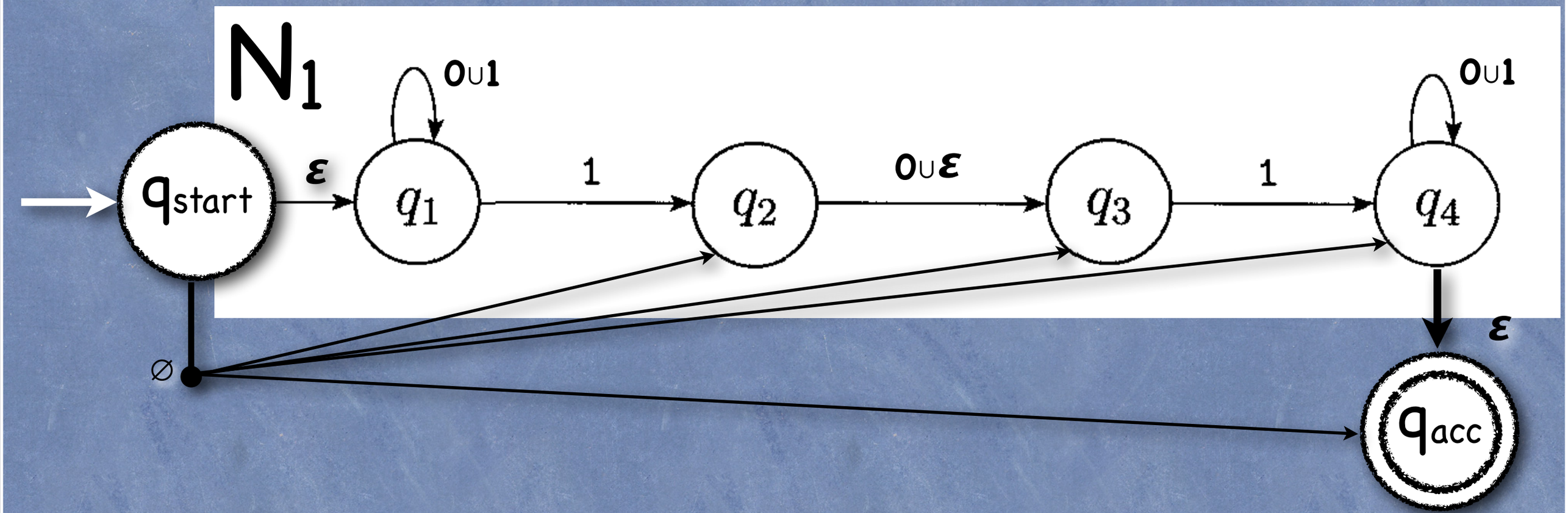
Example NFA→GNFA



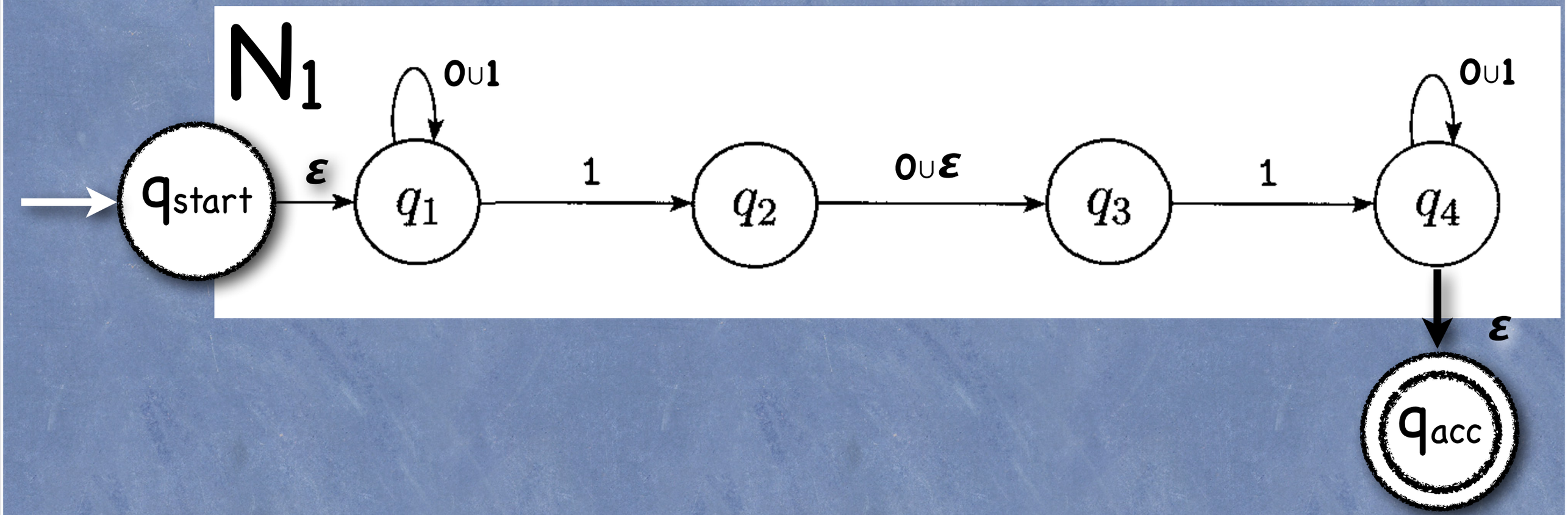
Example NFA→GNFA



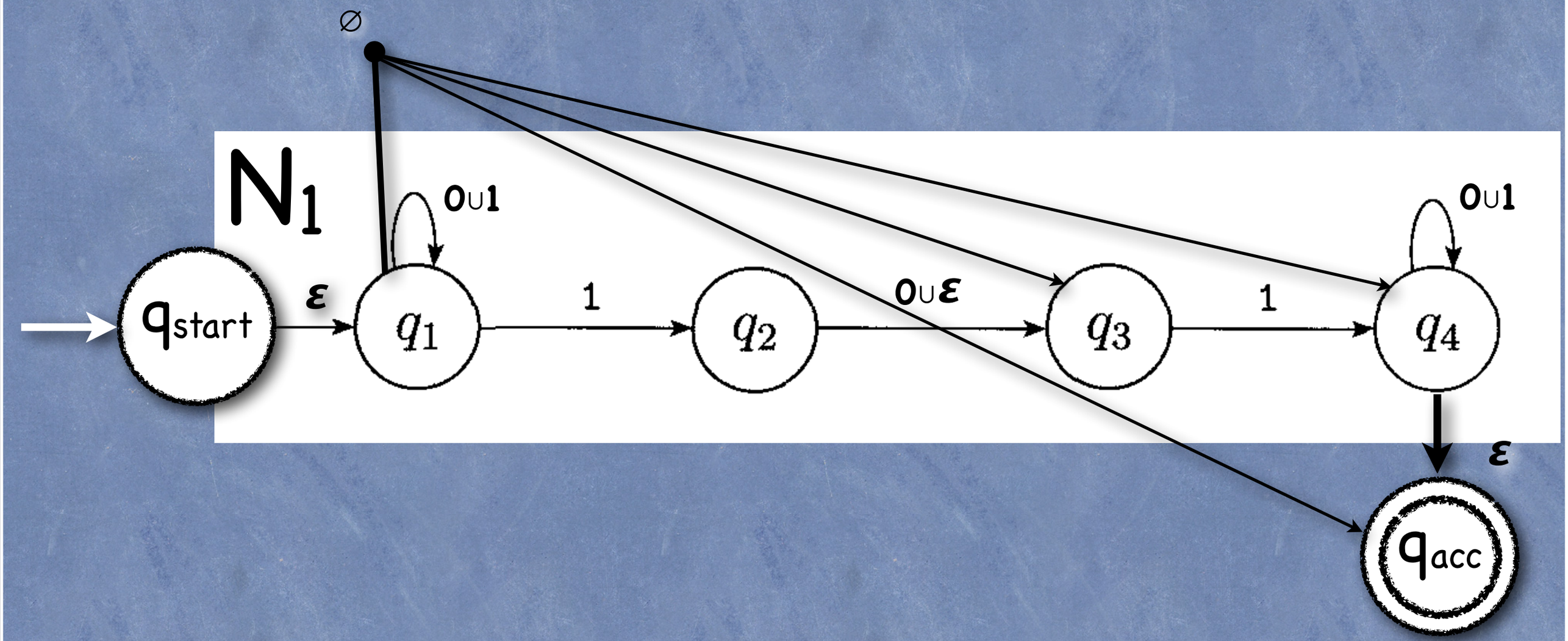
Example NFA→GNFA



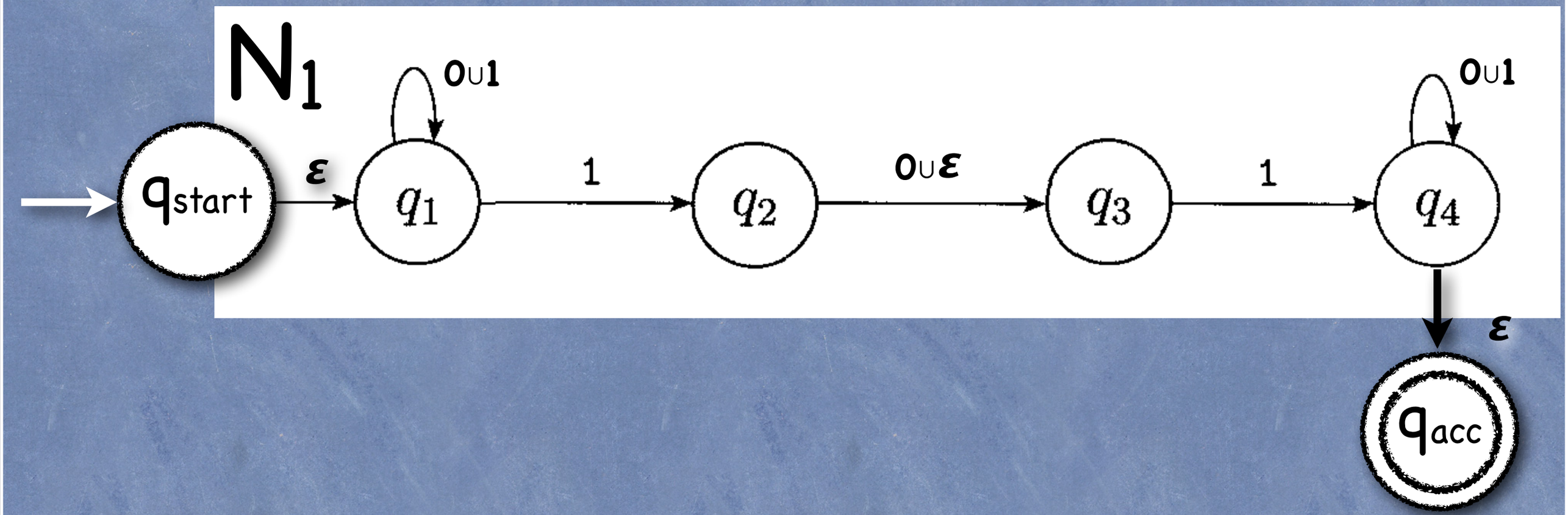
Example NFA→GNFA



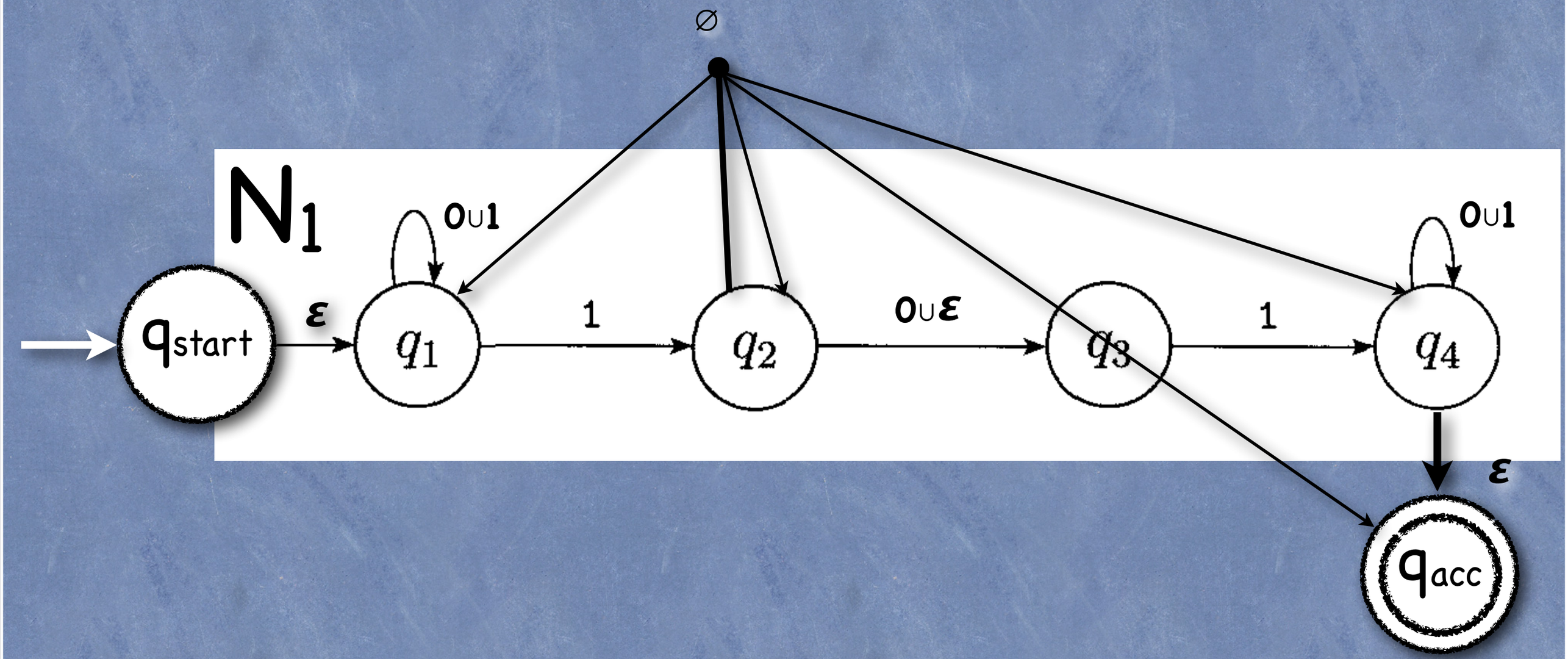
Example NFA→GNFA



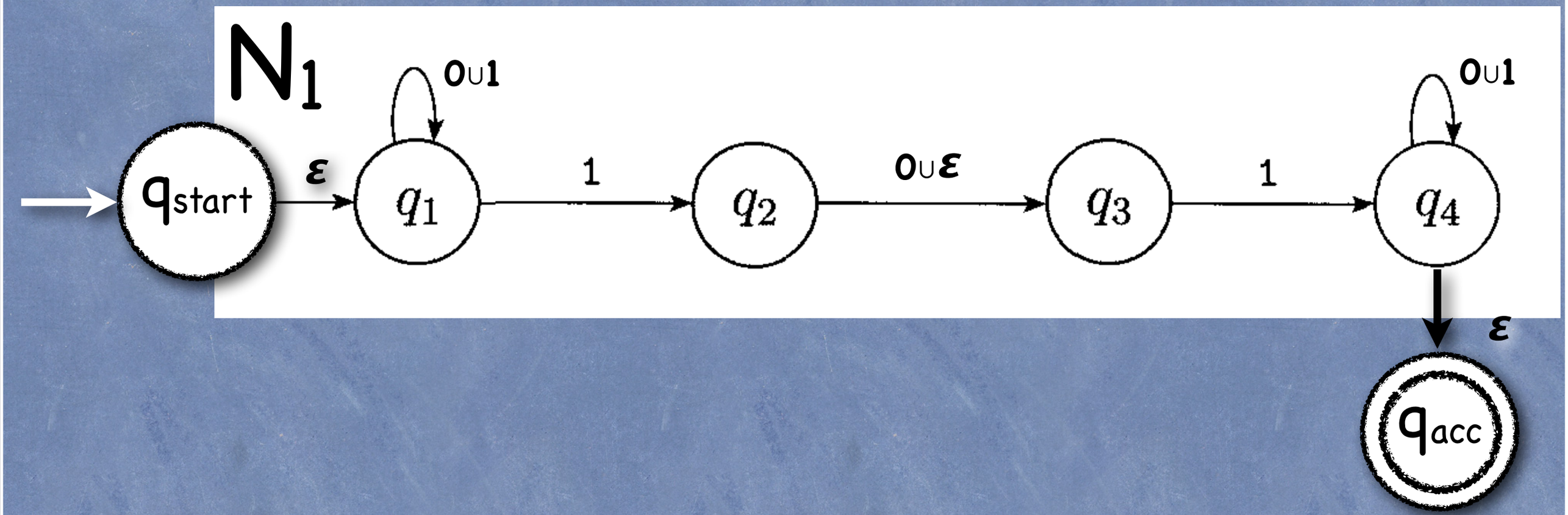
Example NFA→GNFA



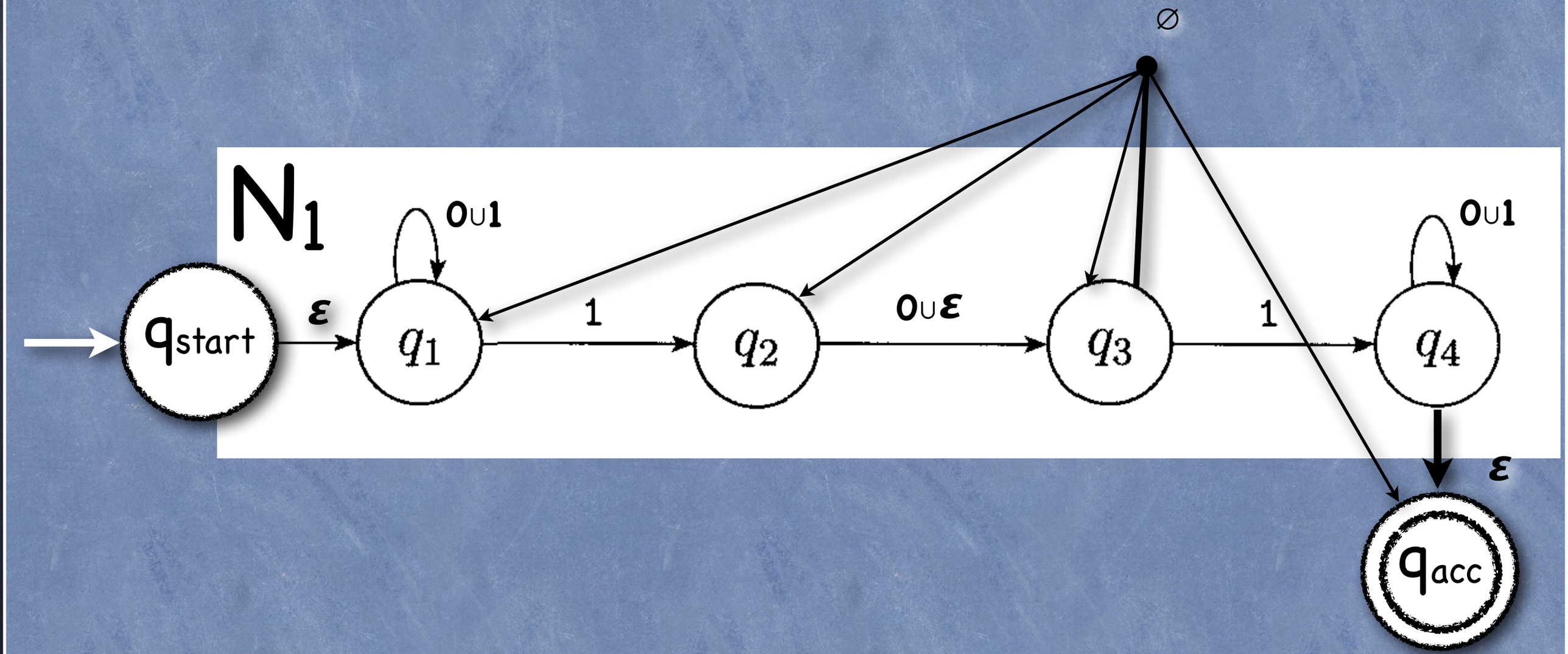
Example NFA→GNFA



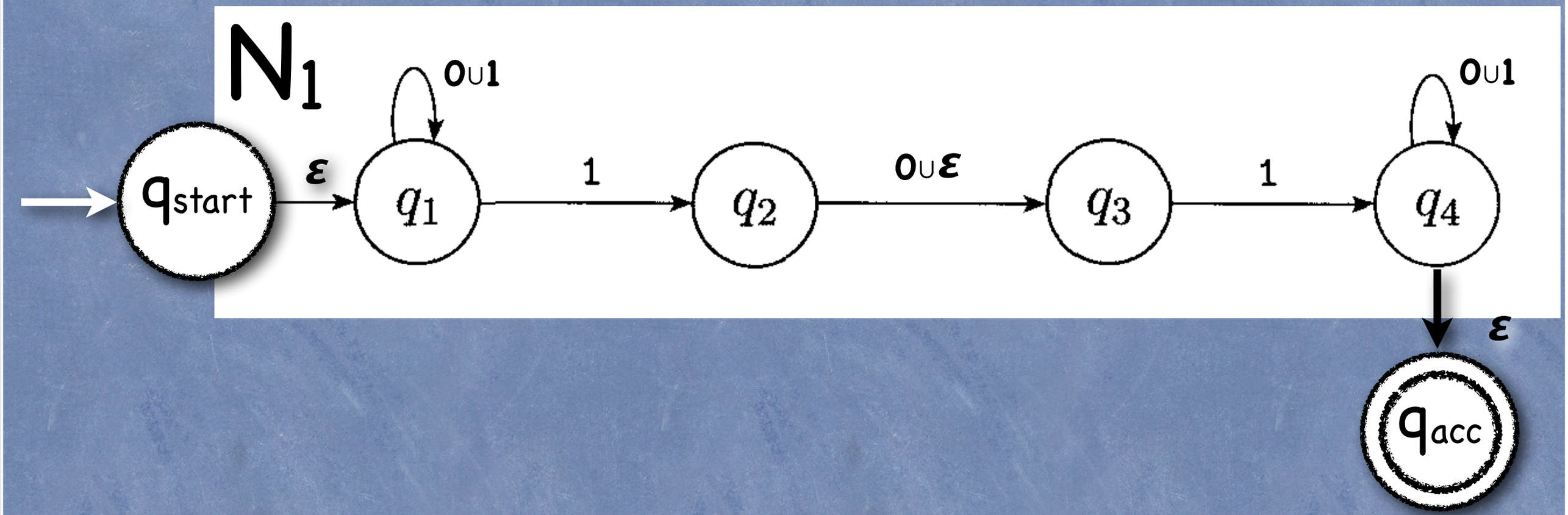
Example NFA→GNFA



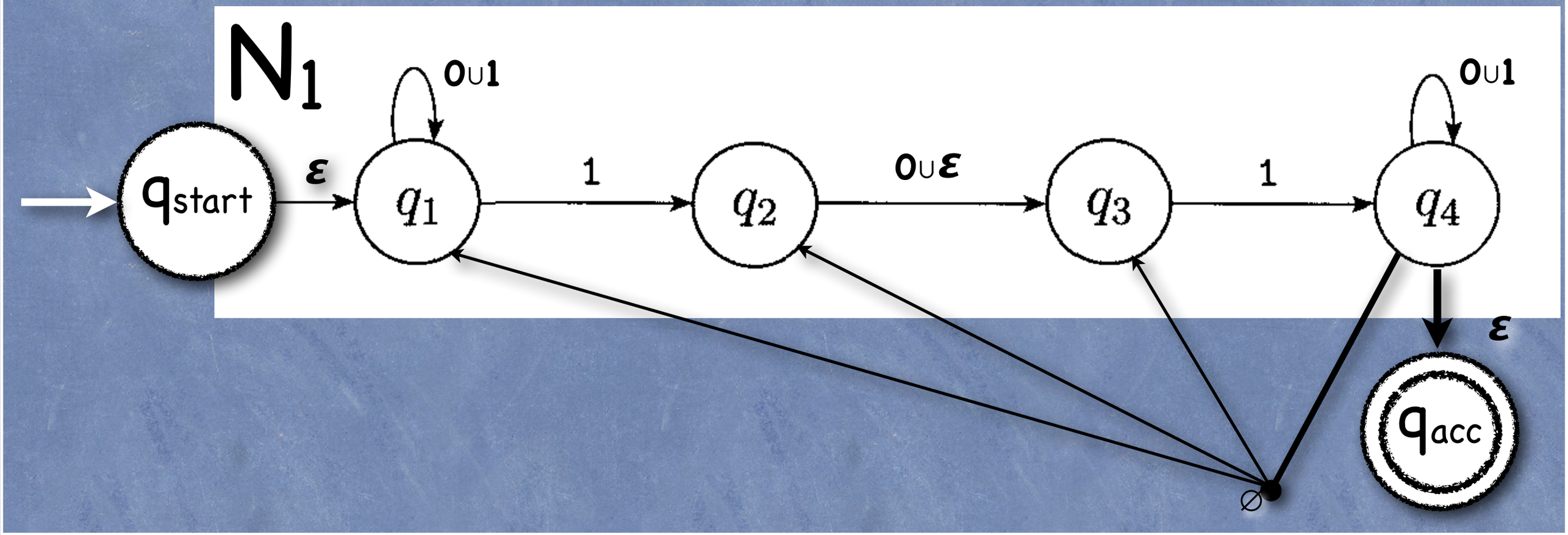
Example NFA→GNFA



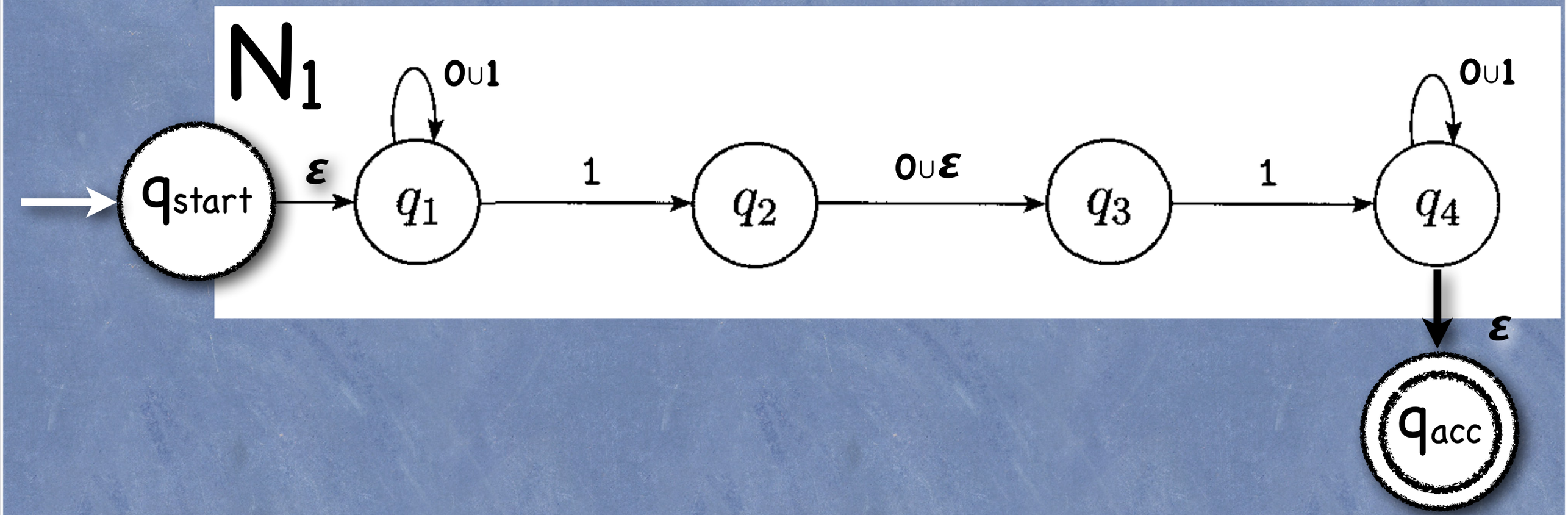
Example NFA→GNFA



Example NFA→GNFA

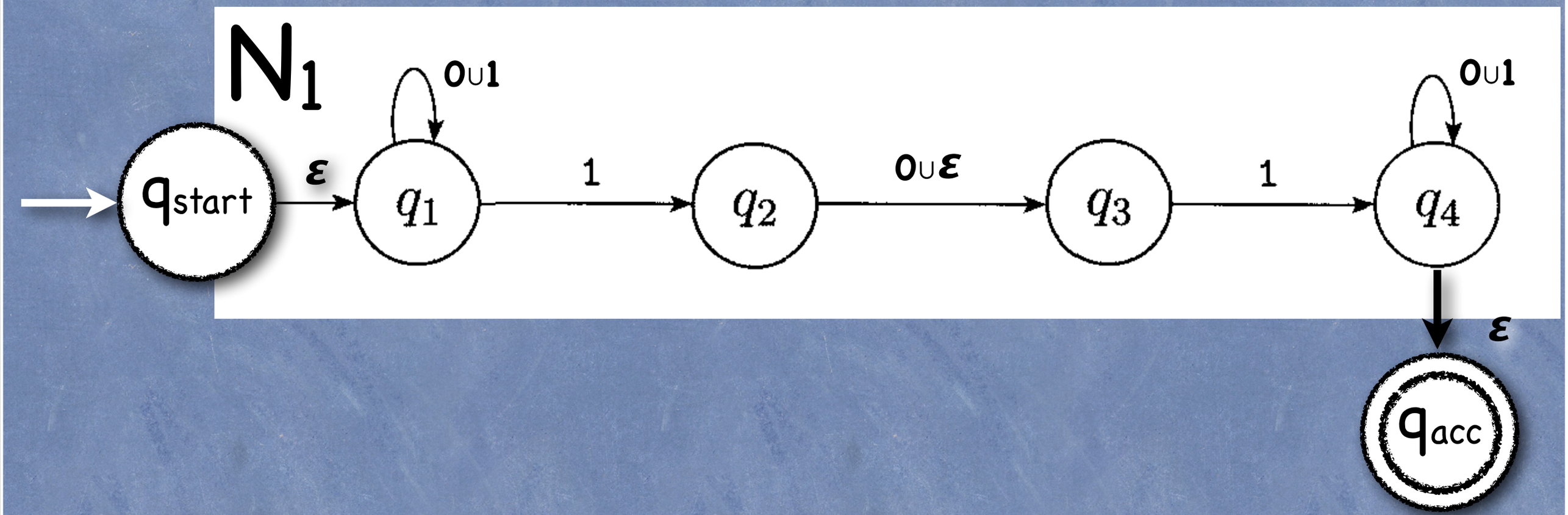


Example NFA→GNFA



Example NFA→GNFA

G_1



DFA → GNFA → Reg. Exp.

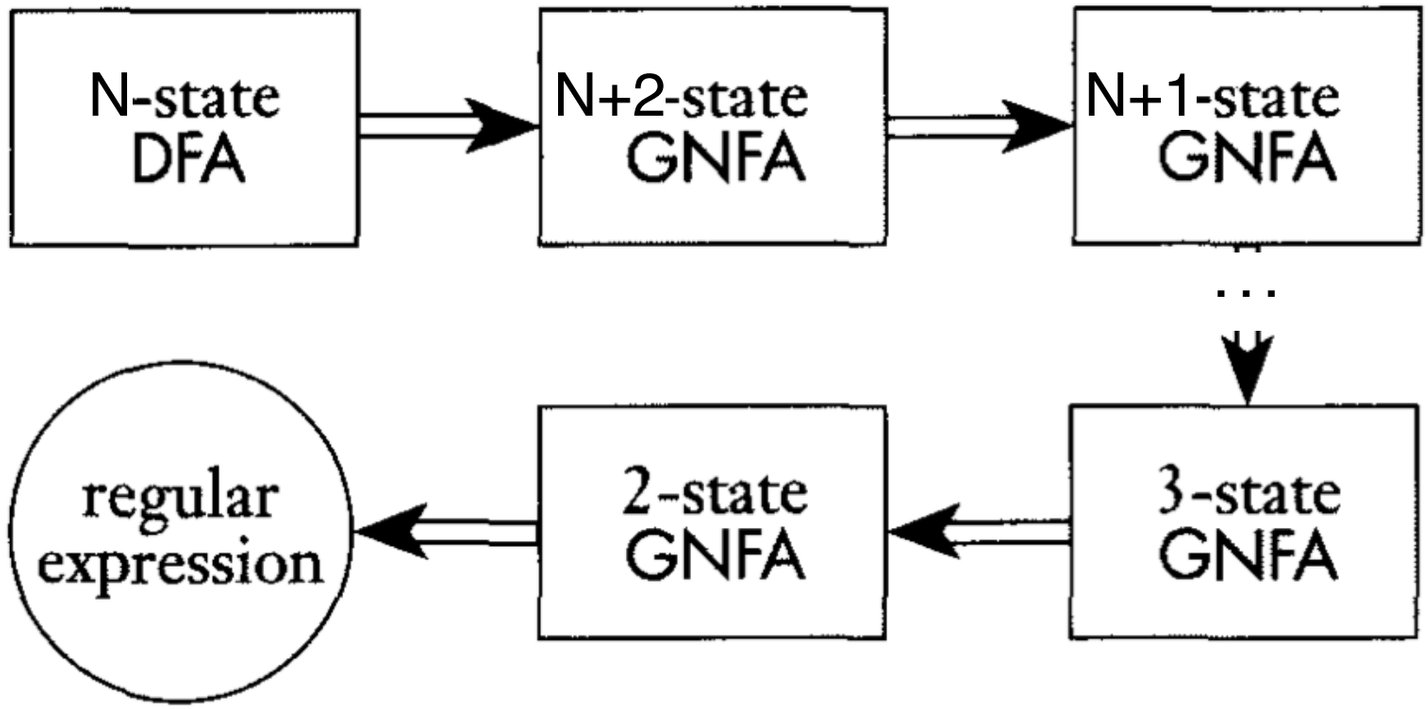


FIGURE 1.62
Typical stages in converting a DFA to a regular expression

Ripping a state

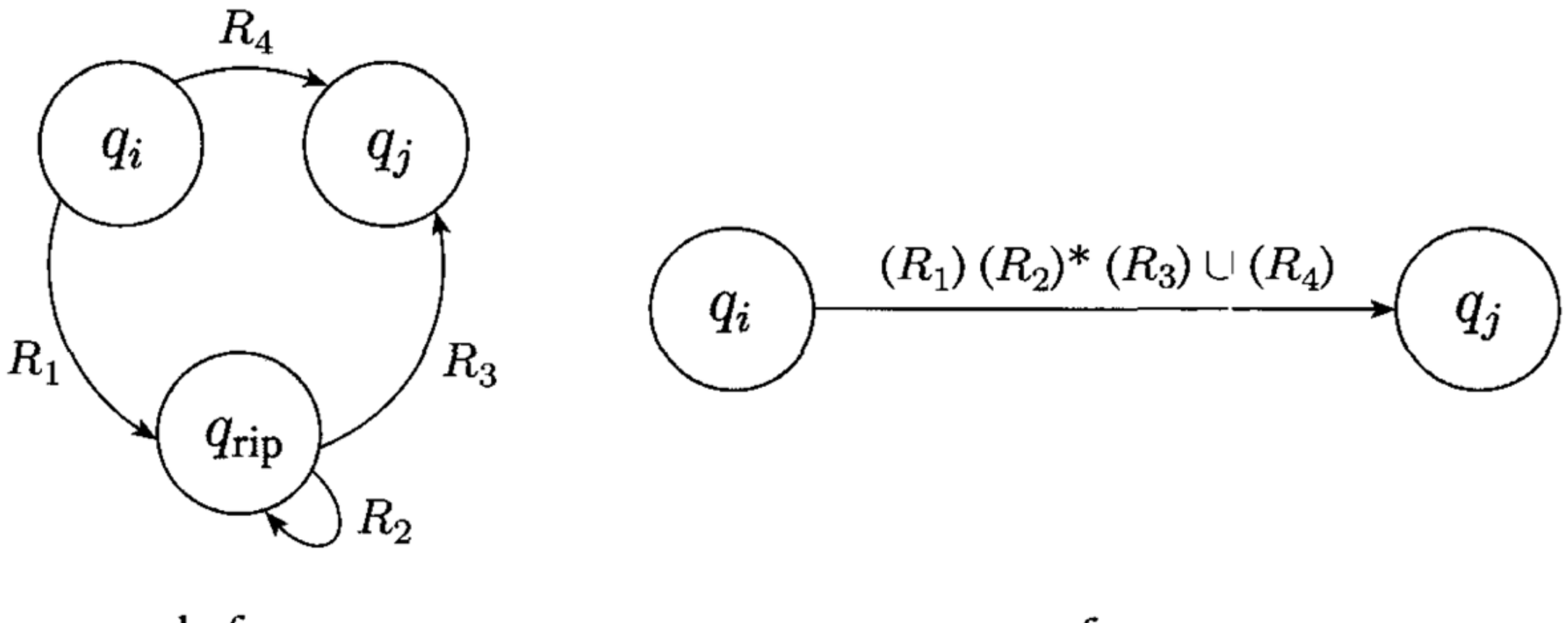
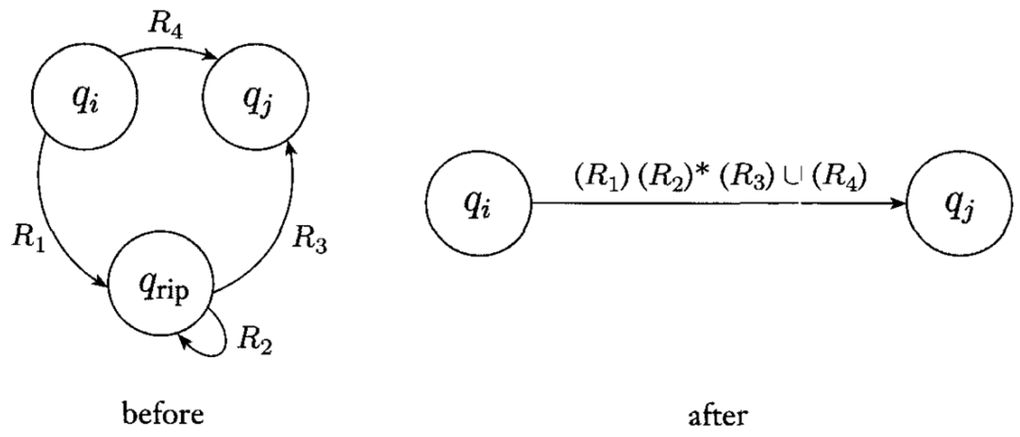


FIGURE 1.63
Constructing an equivalent GNFA with one fewer state

GNFA→ Reg. Exp.



CONVERT(G):

1. Let k be the number of states of G.
2. If $k = 2$, then G must consist of a start state, an accept state, and a single arrow connecting them and labeled with a regular expression R. Return the expression R.
3. If $k > 2$, we select any state $q_{\text{rip}} \in Q$ different from q_{start} and q_{accept} and let G' be the GNFA $(Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}})$, where

$$Q' = Q - \{q_{\text{rip}}\},$$

and for any $q_i \in Q' - \{q_{\text{accept}}\}$ and any $q_j \in Q' - \{q_{\text{start}}\}$ let

$$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4),$$

for $R_1 = \delta(q_i, q_{\text{rip}})$, $R_2 = \delta(q_{\text{rip}}, q_{\text{rip}})$, $R_3 = \delta(q_{\text{rip}}, q_j)$, and $R_4 = \delta(q_i, q_j)$.

4. Compute CONVERT(G') and return this value.

GNFA → Reg. Expression

CLAIM 1.65

For any GNFA G, CONVERT(G) is equivalent to G.

We prove this claim by induction on k, the number of states of the GNFA.

"equivalent" means L(CONVERT(G)) = L(G)

DFA → GNFA → Reg. Exp.

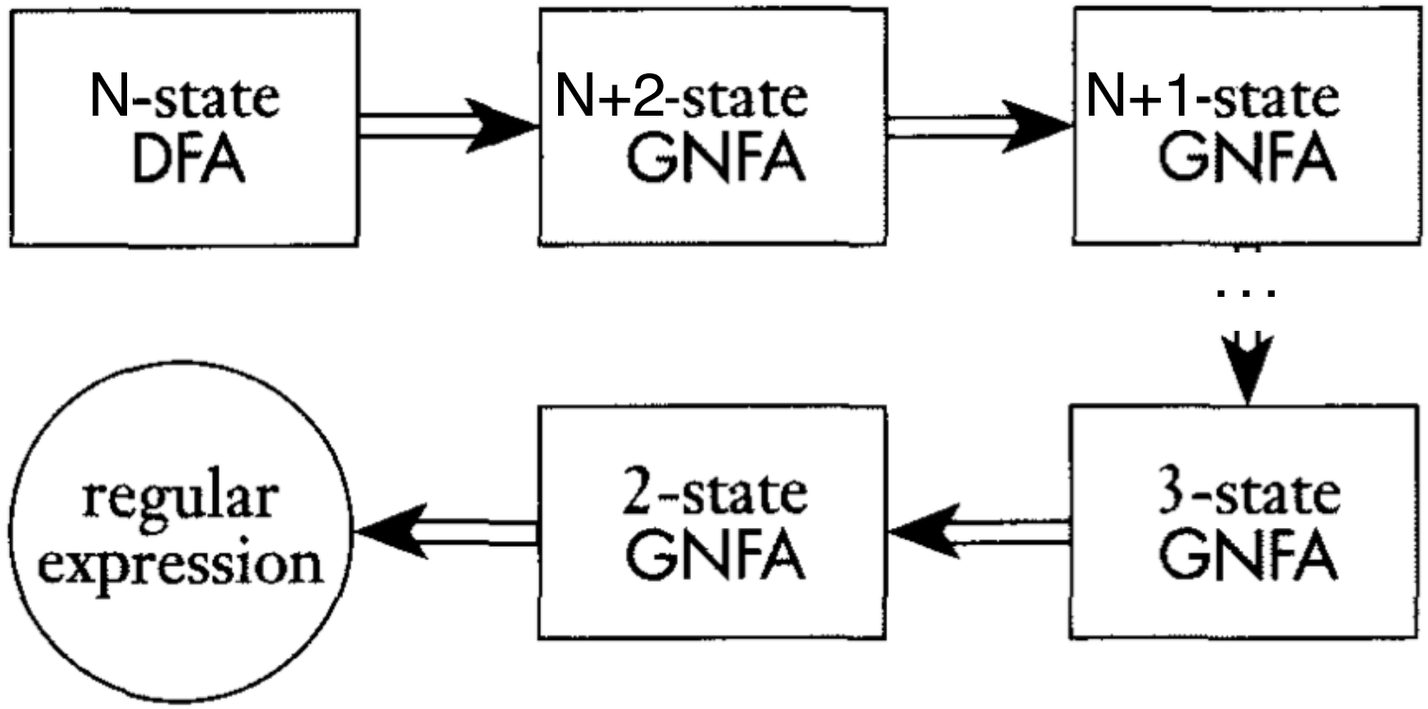


FIGURE 1.62
Typical stages in converting a DFA to a regular expression

Application of the Myhill-Nerode Theorem

Application of the Myhill-Nerode Theorem

Application of the Myhill-Nerode Theorem

Application of the Myhill-Nerode Theorem

Application of the Myhill-Nerode Theorem

Application of the Myhill-Nerode Theorem

Given two regular expressions R and R′ we can find out whether they generate the same regular language or not :

1. From R and R′, compute NFAs N and N′ accepting L(R) and L(R′) (Lemma 1.55).

2. Compute equivalent DFAs M and M′ (Thm 1.39).

3. Using part (b) of Myhill-Nerode we construct minimal DFAs W for M and W′ for M′.

4. L(R)=L(R′) iff W≈W′
   (≈ means "identical up to state renaming").

Regular and non-Regular Languages

footnote 3 page 46:

footnote 3 page 46:

footnote 3 page 46:

footnote 3 page 46:

footnote 3 page 46:

- Let $M_A=(Q_A,\Sigma,\delta_A,q_{0A},F_A)$ be a DFA accepting L_A and $M_B=(Q_B,\Sigma,\delta_B,q_{0B},F_B)$ be a DFA accepting L_B.

- Consider $M_U=(Q_A \times Q_B,\Sigma,\delta_U,(q_{0A},q_{0B}),F_U)$ where

  $\delta_U((q,q'),s) = (\delta_A(q,s), \delta_B(q',s))$ for all q,q',s

  and

  $F_U = (FA \times QB) \cup (QA \times FB)$.

- $L_U = L_A \cup L_B$.

- $F_U = F_A \times F_B$ would yield the **intersection** (and not the **union**) of L_A and L_B.
  This proves that the class of regular languages is also closed under intersection.

NON-Regular Languages

- $B = \{ 0^n1^n \mid n \geq 0 \}$
- C = { w | w contains an equal number of 0's and 1's }
- D = { w | w contains an equal number of occurrences of 01 and 10 as sub-strings }

NON-Regular Languages

NON-Regular

- B = { $0^n 1^n 0^n$ }
- C = { w | w contains an equal number of 0's and 1's }
- D = { w | w contains an equal number of occurrences of 01 and 10 as sub-strings }

NON-Regular Languages

- B = { $0^n 1^n$ | n ≥ 0 }

NON-Regular

- C = { w | w contains an equal number of 0's and 1's }

NON-Regular

- D = { w | w contains an equal number of occurrences of 01 and 10 as sub-strings }

NON-Regular Languages

- B = { $0^n 1^n$ | n ≥ 0 }

NON-Regular

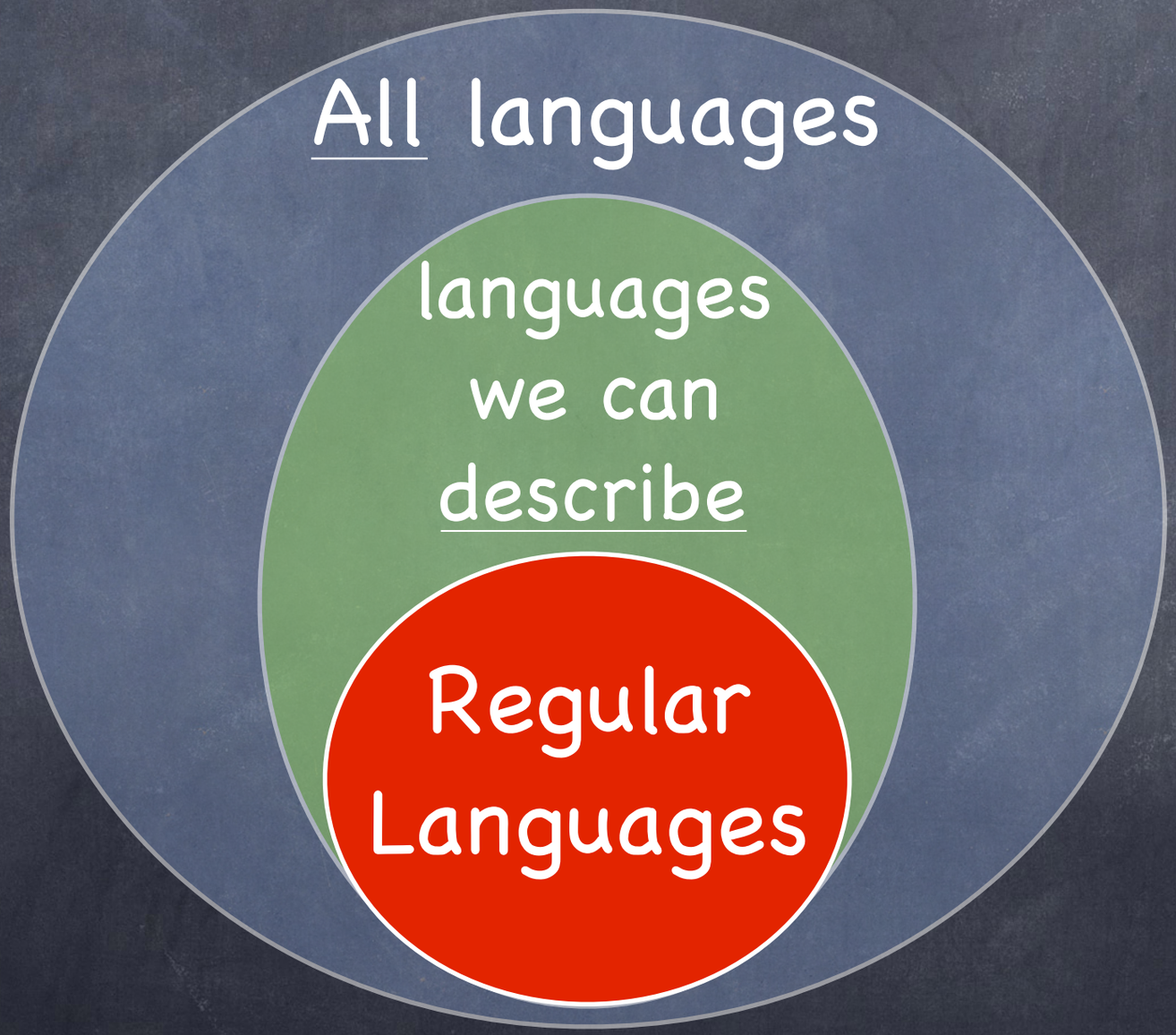
- C = { w | w contains equal number of 0's and 1's }

NON-Regular

- D = { w | w contains equal number of occurrences of 01 and 10 as sub-strings }

Regular

Computability Theory

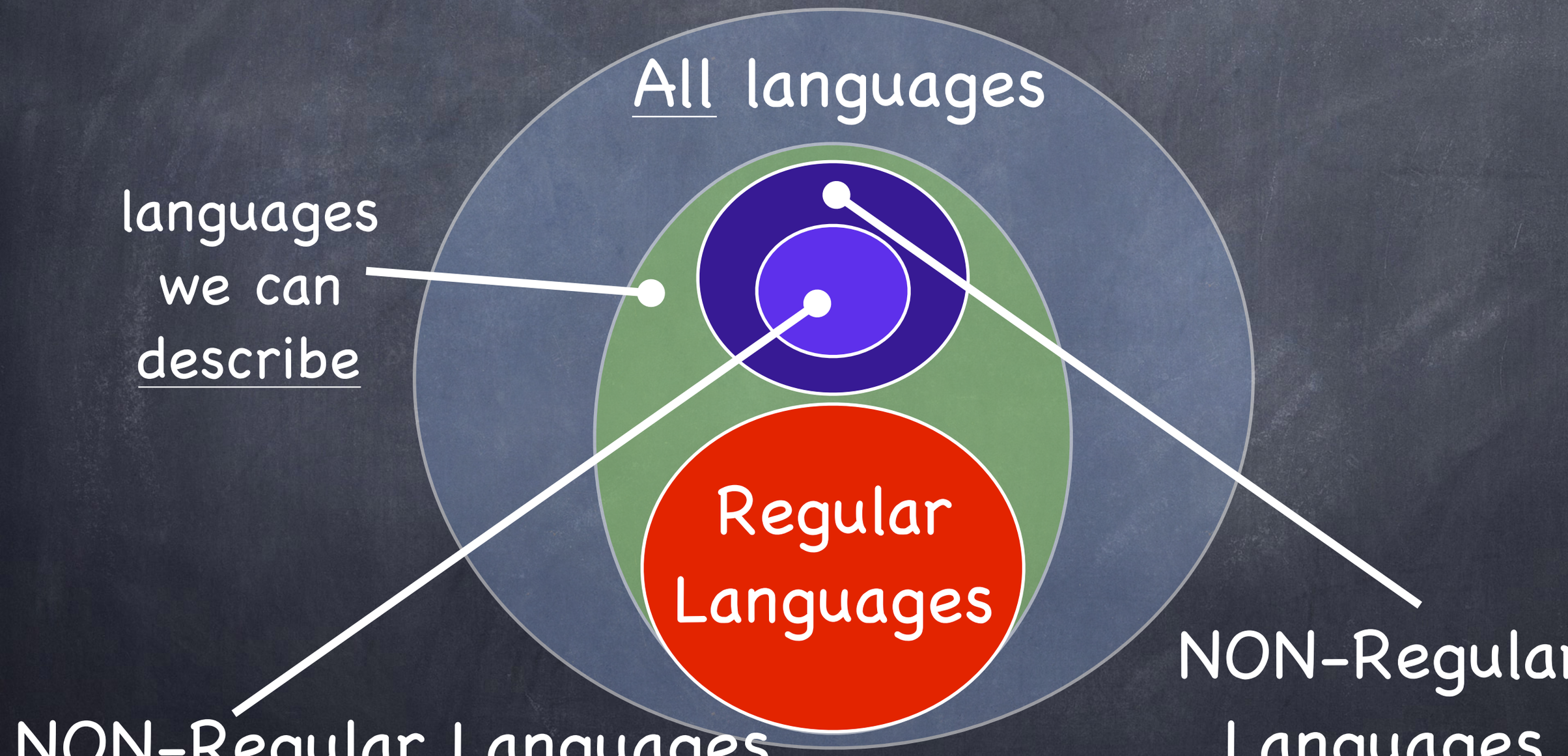


NON-Regular Languages

- Theorem: Some languages are not regular.

Proof idea: all regular languages have certain properties. Some languages provably do not have one of these properties.

Computability Theory



Reductions

- If C is regular then so is B.
- Proof: Regular languages are closed under intersection (see footnote 3 page 46). Define A = L(0*1*). Obviously A is regular. If C was regular then so would C∩A = B.

  QED

- If B is NON-regular then so is C.

$B = \{ 0^n1^n \mid n \geq 0 \}$

$C = \{ w \mid w \text{ contains an equal number of 0's and 1's} \}$

Reductions

- If A is regular then so is A′.
- Regular laguages are closed under complement (see ex. 1.14), intersection, union, concatenation and star. If there exists R, a regular language, such that either $A^c=A'$, $A^*=A'$, $A\cap R=A'$, $A\cup R=A'$, $A\circ R=A'$ or any combinations of these operations then A′ is regular as long as A is.
- If A′ is NON-regular then so is A.

Simple Reductions

- If A^* is NON-regular then so is A.
- If A is NON-regular then so is A^C.
- If A is NON-regular then so is A^R.

Complex Reductions

- Let $A' = (A \cup R) \cap (A^c \cup R')$ (R,R' regular)
- Let $A' = ((A^c \cap R) \cup (A^* \cap R')) \circ R''$ (R,R',R'' regular)
- Let $A' = (A \circ R) \cap (A^c \circ R')$ (R,R' regular)
- If A' is NON-regular then so is A.

NON-Regular Languages

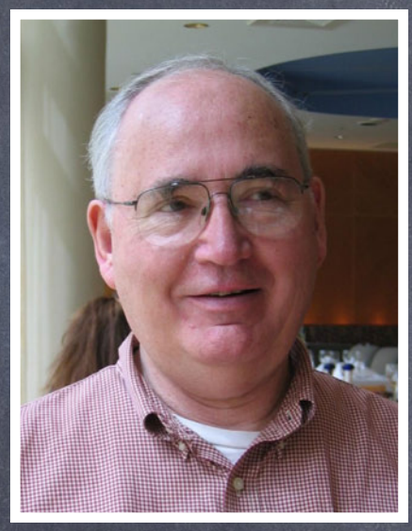
- Theorem: Some languages are not regular.

  Proof idea: all regular languages have certain properties. Some languages provably do not have one of these properties.

- Example: A property of all regular languages = the Pumping Lemma.

Pumping Lemma

Michael Rabin



Dana Scott

NON-Regular Languages

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

- Application: any language that does not satisfy the pumping lemma is non-regular.
- Note however that some non-regular languages DO satisfy the Pumping Lemma...

Pumping Lemma

- If |xyz|>number-of-states then q_9 exists...

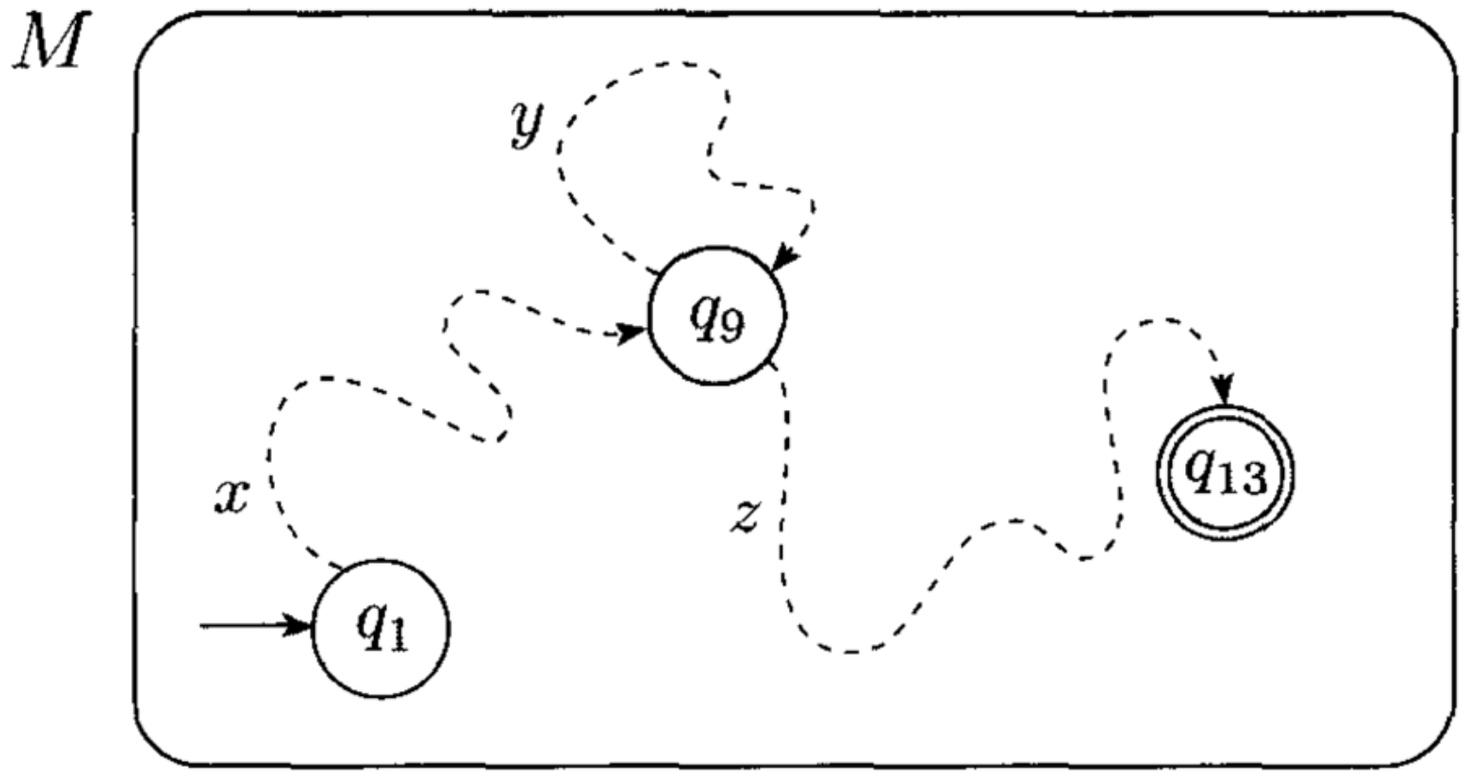


FIGURE 1.72
Example showing how the strings x, y, and z affect M

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

$A \in \mathbb{REG} \Rightarrow$

$\exists p \forall s \in A,\ |s| \geq p,\ \exists xyz = s$ st 1,2,3=true.

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

$A \in \mathbb{REG} \Rightarrow$

$\exists p \forall s \in A,\ |s| \geq p,\ \exists xyz = s$ st 1,2,3=true.

$\forall p \exists s \in A,\ |s| \geq p,\ \forall xyz = s$ [1 or 2 or 3 = false].

$\Rightarrow A \notin \mathbb{REG}$

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

$A \in \mathbb{REG} \Rightarrow$

$\exists p \forall s \in A,\ |s| \geq p,\ \exists xyz = s$ st 1,2,3=true.

$\forall p \exists s \in A,\ |s| \geq p,\ \forall xyz = s$ st 2,3=true [1=false].

$\Rightarrow A \notin \mathbb{REG}$

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

$A \in \mathbb{REG} \Longrightarrow$

$\exists p \forall s \in A,\ |s| \geq p,\ \exists xyz = s$ st 1,2,3=true.

$\forall p \exists s \in A,\ |s| \geq p,\ \forall xyz = s$ st 2,3=true [1=false].

$\Longrightarrow A \notin \mathbb{REG}$

$\forall p \exists s \in A,\ |s| \geq p,\ \forall xyz = s$ s.t. $|y| > 0, |xy| \leq p$,
then $\exists i \geq 0$ s.t. $s' = xy^iz \notin A$.

$\Longrightarrow A \notin \mathbb{REG}$

Application of the Pumping Lemma

- $B = \{ 0^n1^n \mid n \geq 0 \}$ is NON-Regular.
- Assume B is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take n=p and set $s = 0^p1^p \in B$. Then by 3. xy contains only zeros. Therefore if we pump even once to obtain $s' = xyyz = 0^q1^p$ it will contain more zeros than ones (q>p) : a string s' not in B. Thus B is non-regular.

$$\forall p \exists s \in B,\ |s| \geq p,\ \forall xyz = s \text{ s.t. } |y| > 0, |xy| < p,$$
$$\text{then } \exists i \geq 0 \text{ s.t. } s' = xy^i z \notin B.$$
$$\Longrightarrow B \notin \mathbb{REG}$$

- $B = \{ 0^n 1^n \mid n \geq 0 \}$ is NON-Regular.
- Assume B is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take n=p and set $s = 0^p 1^p \in B$. Then by 3. xy contains only zeros. Therefore if we pump even once to obtain $s' = xyyz = 0^q 1^p$ it will contain more zeros than ones (q>p) : a string s' not in B. Thus B is non-regular.

Application of the Pumping Lemma

- $F = \{ ww \mid w \in \Sigma^* \}$ is NON-Regular.
- Assume F is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take $s=0^p10^p1 \in F$. Then by 3. xy contains only zeros. Therefore if we pump even once to obtain $s'=xyyz$ it will contain more zeros before the first one than after the first one : a string s' not in F. Thus F is non-regular.

$$\forall p \exists s \in F,\ |s| \geq p,\ \forall xyz = s \text{ s.t. } |y| > 0, |xy| < p,$$
$$\text{then } \exists i \geq 0 \text{ s.t. } s' = xy^i z \notin F.$$
$$\Longrightarrow F \notin \mathbb{REG}$$

- $F = \{ ww \mid w \in \Sigma^* \}$ is NON-Regular.
- Assume F is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take $s = 0^p 1 0^p 1 \in F$. Then by 3. xy contains only zeros. Therefore if we pump even once to obtain $s' = xyyz$ it will contain more zeros before the first one than after the first one : a string s' not in F. Thus F is non-regular.

Application of the Pumping Lemma

- $E = \{ 0^i1^j \mid i>j\geq 0 \}$ is NON-Regular.
- Assume E is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take i=p+1, j=p and obtain $s=0^{p+1}1^p \in E$. Then by 3. xy contains only zeros.

$$\forall p \exists s \in E,\ |s| \geq p,\ \forall xyz = s \text{ s.t. } |y| > 0, |xy| < p,$$
$$\text{then } \exists i \geq 0 \text{ s.t. } s' = xy^i z \notin E.$$
$$\Longrightarrow E \notin \mathbb{REG}$$

- $E = \{ 0^i 1^j \mid i > j \geq 0 \}$ is NON-Regular.
- Assume E is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take i=p+1, j=p and obtain $s = 0^{p+1}1^p \in E$. Then by 3. xy contains only zeros.

Application of the Pumping Lemma

- $E = \{ 0^i1^j \mid i>j\geq 0 \}$ is NON-Regular.
- Therefore if we pump up to obtain $s'=xyyyz=0^k1^j$, $k>i$ it will contain even more zeros than ones, which is still a string s' in E. If we pump down however $s''=xz$, the number of zeros will become smaller or equal to the number of ones: an s'' not in E. Thus E is non-regular.

$$\forall p \exists s \in E,\ |s| \geq p,\ \forall xyz = s \text{ s.t. } |y| > 0, |xy| < p,$$
$$\text{then } \exists i \geq 0 \text{ s.t. } s'' = xy^i z \notin E.$$
$$\Longrightarrow E \notin \mathbb{REG}$$

- $E = \{ 0^i 1^j \mid i > j \geq 0 \}$ is NON-Regular.
- Therefore if we pump up to obtain $s' = xyyz = 0^k 1^j$, $k > i$ it will contain even more zeros than ones, which is still a string s' in E. If we pump down however $s'' = xz$, the number of zeros will become smaller or equal to the number of ones: an s'' not in E. Thus E is non-regular.

NON-Application of the Pumping Lemma

1.54 Consider the language $F = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \geq 0 \text{ and if } i = 1 \text{ then } j = k\}$.

a. Show that F is not regular.

b. Show that F acts like a regular language in the pumping lemma. In other words, give a pumping length p and demonstrate that F satisfies the three conditions of the pumping lemma for this value of p.

c. Explain why parts (a) and (b) do not contradict the pumping lemma.

- c. The Pumping Lemma says: if A is regular then 1., 2. and 3. are satisfied. It does not say: if A is not regular then 1., 2. or 3. is not satisfied... We can only conclude the opposite: if 1., 2. or 3. is not satisfied then A is not regular...

Application of the Pumping Lemma

- D = { 1^{n^2} | n≥0 } is NON-Regular.

- Assume D is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take n=p and obtain $s=1^{p^2}$. Let $i=|y|\leq p$. If we pump up we get $s''=xyyz=1^{p^2+i}$. Is it possible that both p^2 and p^2+i be perfect squares ? No! The next square after p is

$$(p+1)^2 = p^2+2p+1 > p^2+p+1 > p^2+i$$

proving that s" is not in D. So D is non-regular.

$$\forall p \exists s \in D,\ |s| \geq p,\ \forall xyz = s \text{ s.t. } |y| > 0, |xy| < p,$$
$$\text{then } \exists i \geq 0 \text{ s.t. } s'' = xy^i z \notin D.$$
$$\Longrightarrow D \notin \mathbb{REG}$$

- $D = \{ 1^{n^2} \mid n \geq 0 \}$ is NON-Regular.
- Assume D is regular. Then by the pumping Lemma there exists a pumping length p with properties 1., 2. and 3. satisfied. Take n=p and obtain $s=1^{p^2}$. Let $i=|y| \leq p$. If we pump up we get $s''=xyyz=1^{p^2+i}$. Is it possible that both p^2 and p^2+i be perfect squares ? No! The next square after p is

$$(p+1)^2 = p^2+2p+1 > p^2+p+1 > p^2+i$$

proving that s" is not in D. So D is non-regular.

All languages

Computability Theory

Languages we can describe

Decidable Languages

Context-free Languages

Regular Languages

NON-Regular Languages via Pumping Lemma or MN Theorem

NON-Regular Languages via Reductions

COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

LECTURE 14a :
MIDTERM REVIEW