Theory of Computing
SE-205

Lecture-2

Extended Transition Function

We describe the effect of a string of inputs on a DFA by extending $\hat{\delta}$ to a state and a string.

Induction on length of string.

Basis: $\hat{\delta}(q, \epsilon) = q$

Suppose w is a string where w=xa.

w=1101 is broken into x=110 and a=1.

Induction: $\hat{\delta}(q,w) = \delta(\hat{\delta}(q,x),a)=r$

Extended Transition Function..

	0	1
A	A	B
B	A	C
C	C	C

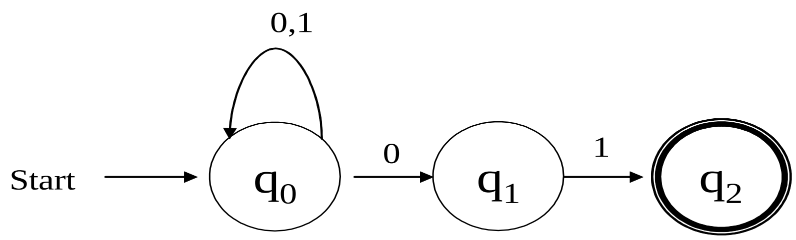
$$\hat{\delta}(B,011) = \delta(\hat{\delta}(B,01),1) = \delta(\delta(\delta(B,0),1),1) =$$

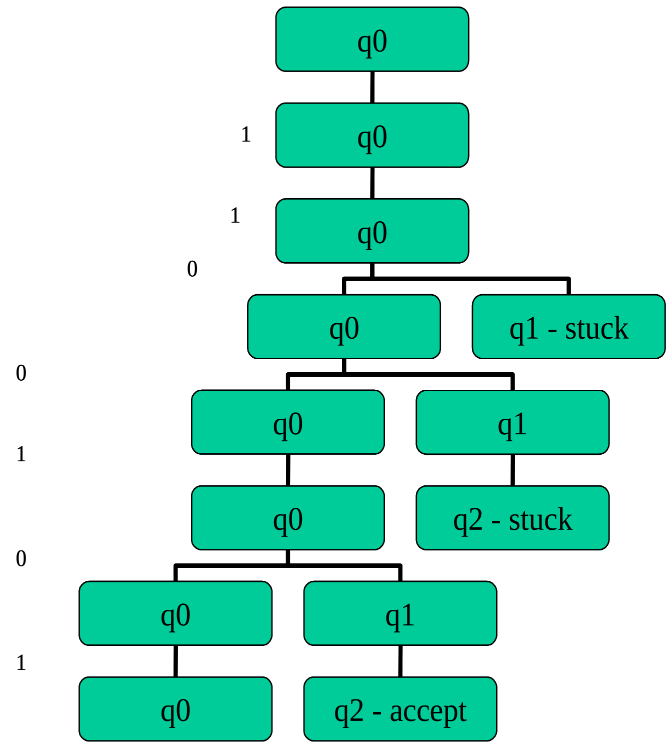
$$\delta(\delta(A,1),1) = \delta(B,1) = C$$

NFA

- A *nondeterministic finite automaton* has the ability to be in several states at once.
- Transitions from a state on an input symbol can be to any set of states.

- This NFA accepts only those strings that end in 01
- Running in "parallel threads" for string 1100101





Language of an NFA

- An NFA accepts *w* if *there exists at least one* path from the start state to an accepting (or final) state that is labeled by *w*
- $L(N) = \{ w \mid \delta(q_0,w) \cap F \neq \Phi \}$

Extended transition function for NFA

- Basis: $\hat{\delta}(q,\varepsilon) = \{q\}$

Induction: Suppose w is a string where $w=xa$.

- Let $\hat{\delta}(q,x) = \{p_1, p_2 \ldots, p_k\}$
- $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, r_3, \ldots, r_m\}$
- $\hat{\delta}(q, w) = \{r_1, r_2, r_3, \ldots r_m\}$

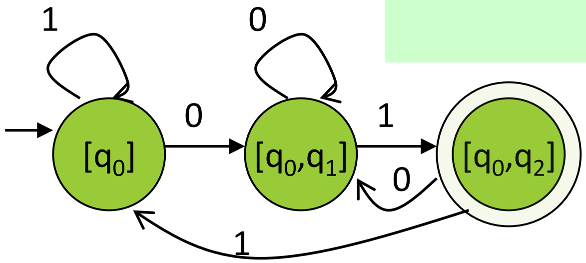
NFA to DFA by Subset Construction

- Given $N = \{Q_N, \sum, \delta_N, q_0, F_N\}$
- Goal: Build $D = \{Q_D, \sum, \delta_D, \{q_0\}, F_D\}$ s.t. $L(D)=L(N)$
- Construction:
  1. Q_D= all subsets of Q_N (i.e., power set)
  2. F_D=set of subsets S of Q_N s.t. $S \cap F_N \neq \Phi$
  3. δ_D: for each subset S of Q_N and for each input symbol a in $\sum$:
     - $\delta_D(S,a) = \bigcup_{p \text{ in } s} \delta_N(p,a)$

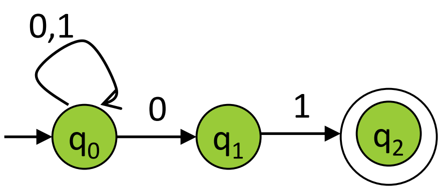
NFA to DFA construction: Example

$L = \{w \mid w \text{ ends in } 01\}$

<u>Idea:</u> To avoid enumerating all of power set, do "lazy creation of states"



<u>NFA:</u>



δ_N	0	1
$\rightarrow q_0$	$\{q_0,q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$*q_2$	$\varnothing$	$\varnothing$

<u>DFA:</u>

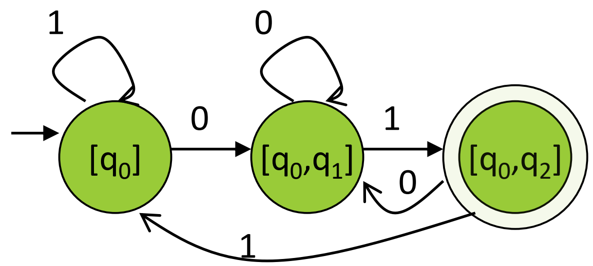
δ_D	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\rightarrow\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
~~$\{q_1\}$~~	~~$\varnothing$~~	~~$\{q_2\}$~~
~~$*\{q_2\}$~~	~~$\varnothing$~~	~~$\varnothing$~~
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
~~$*\{q_0,q_2\}$~~	~~$\{q_0,q_1\}$~~	~~$\{q_0\}$~~
~~$*\{q_1,q_2\}$~~	~~$\varnothing$~~	~~$\{q_2\}$~~
~~$*\{q_0,q_1,q_2\}$~~	~~$\{q_0,q_1\}$~~	~~$\{q_0,q_2\}$~~

δ_D	0	1
$\rightarrow\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
$*\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$

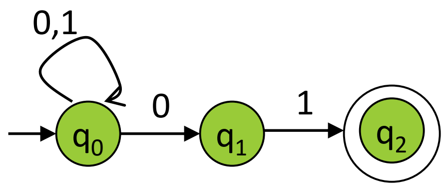
0. Enumerate all possible subsets
1. Determine transitions
2. Retain only those states reachable from $\{q_0\}$

NFA to DFA: Repeating the example using *LAZY CREATION*

L = {w | w ends in 01}



NFA:



δ_N	0	1
$\rightarrow q_0$	$\{q_0,q_1\}$	$\{q_0\}$
q_1	Ø	$\{q_2\}$
$*q_2$	Ø	Ø

→

DFA:

δ_D	0	1
$[q_0]$	$[q_0,q_1]$	$[q_0]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_2]$
$*[q_0,q_2]$	$[q_0,q_1]$	$[q_0]$

Main Idea:

Introduce states as you go (on a need basis)

FA with ε-Transitions

- We can allow explicit ε-transitions in finite automata
  - i.e., a transition from one state to another state without consuming any additional input symbol
  - Explicit ε-transitions between different states introduce non-determinism.
  - Makes it easier sometimes to construct NFAs
  - This means that a transition is allowed to occur without reading in a symbol.

***Definition:* ε -NFAs are those NFAs with at least one explicit ε-transition defined.**

- ε -NFAs have one more column in their transition table

Transition function δ is now a function that takes as arguments:

- A state in Q and
- A member of $\Sigma \cup \{\varepsilon\}$; that is, an input symbol or the symbol ε. We require that ε not be a symbol of the alphabet Σ to avoid any confusion.

ε-Transitions

Use of e-transitions

We allow the automaton to accept the empty string e.

This means that a transition is allowed to occur without reading in a symbol.

The resulting NFA is called e-NFA.

It adds “programming (design) convenience” (more intuitive for use in designing FA’s)

Example # 1: ε-NFA

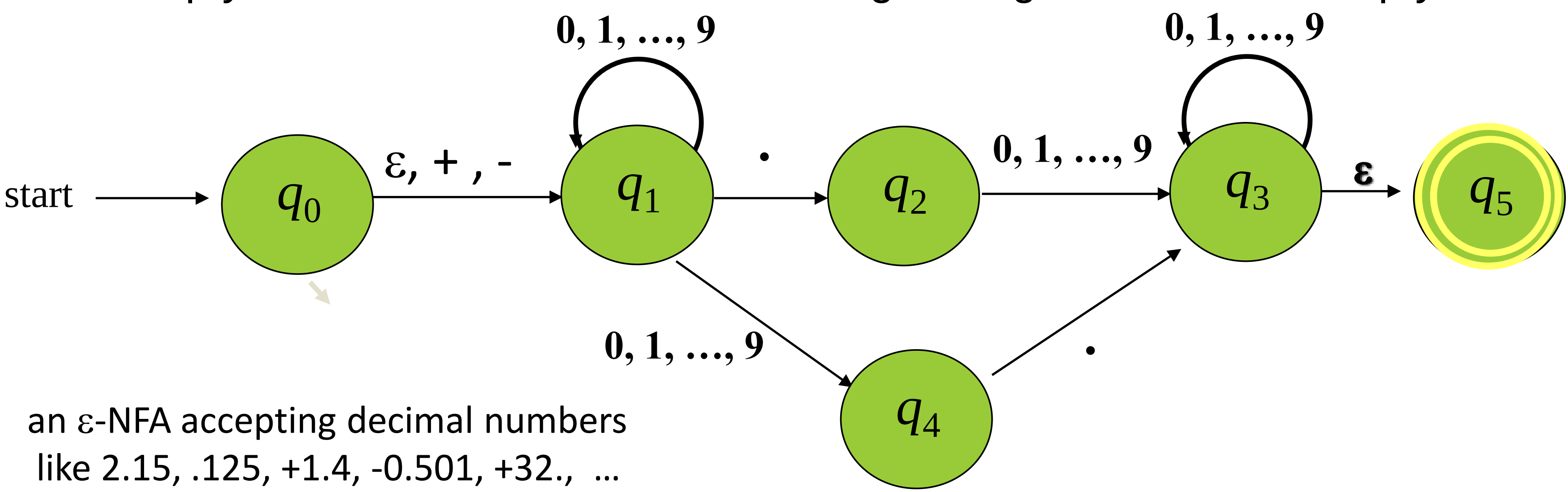
Example: Draw a ε-NFA that accepts decimal numbers consisting of

1. An optional + or – sign

2. A string of digits

3. A decimal point, and

4. Another string of digits. Either this string of digits or the string (2) can be empty, but at least one of the two strings of digit must be nonempty.



an ε-NFA accepting decimal numbers like 2.15, .125, +1.4, -0.501, +32., …

Formal Notation for an ε-NFA

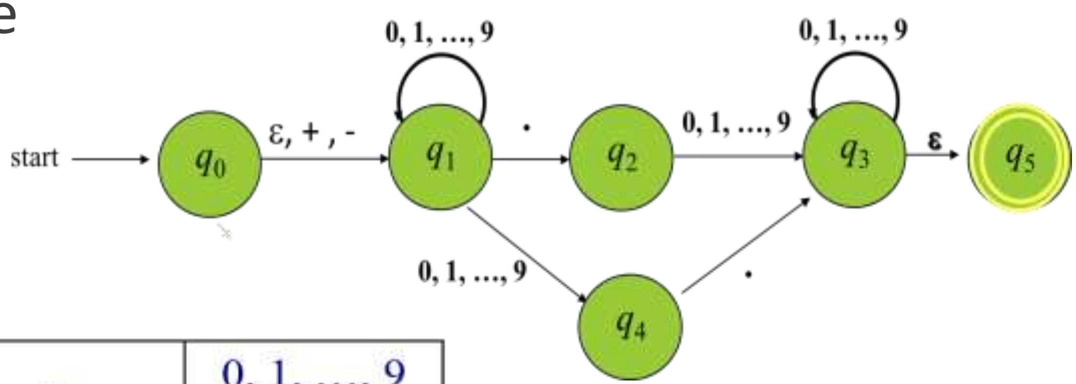
Formal Notation for an e-NFA

E = ({q0, q1, ..., q5}, {., -,+, **ε**, 0, 1, ..., 9}, δ, q0, {q5})

The transitions, e.g., include

$\delta(q_0, \boldsymbol{\varepsilon}) = \{q1\}$

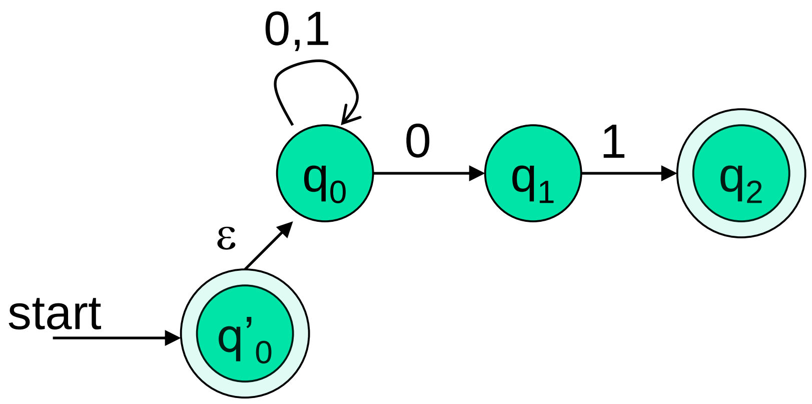
$\delta(q_1, \boldsymbol{\varepsilon}) = \varnothing$



	ε	+, −	.	0, 1, ..., 9
q_0	$\{q_1\}$	$\{q_1\}$	φ	φ
q_1	φ	φ	$\{q_2\}$	$\{q_1, q_4\}$
q_2	φ	φ	φ	$\{q_3\}$
q_3	$\{q_5\}$	φ	φ	$\{q_3\}$
q_4	φ	φ	$\{q_3\}$	φ
q_5	φ	φ	φ	φ

Example #2: ε-NFA..

L = {w | w is empty, <u>or</u> if non-empty will end in 01}



δ_E	0	1	ε-closure
$\rightarrow$ $*q'_0$	Ø	Ø	$\{q'_0, q_0\}$
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	Ø	$\{q_2\}$	$\{q_1\}$
$*q_2$	Ø	Ø	$\{q_2\}$

ECLOSE(q'_0)

ECLOSE(q_0)

ECLOSE(q_1)

ECLOSE(q_2)

- ε-closure of a state q, ***ECLOSE(q)***, is the set of all states (including itself) that can be *reached* from q by repeatedly making an arbitrary number of ε-transitions.

Epsilon-Closures

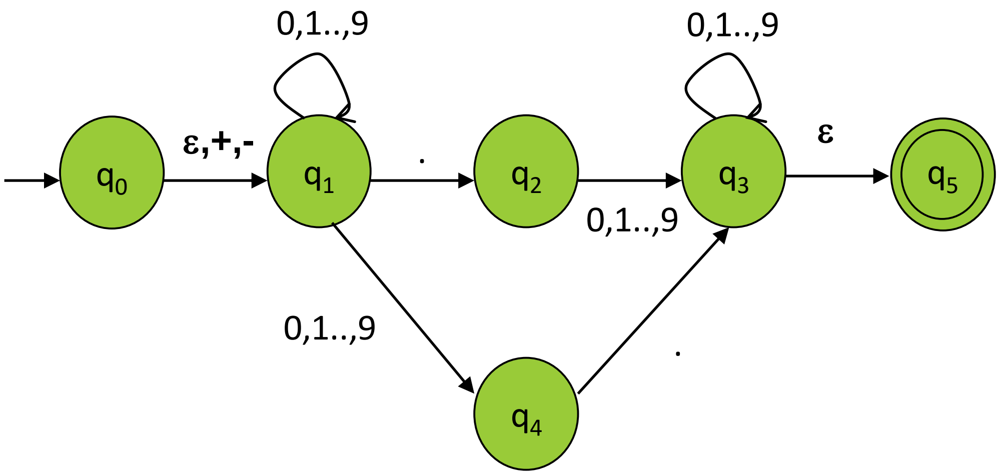
We **ε-close** a state *q* by following all transitions out of *q* that are labeled **ε.**

Basis: state q is in ECLOSE(q).

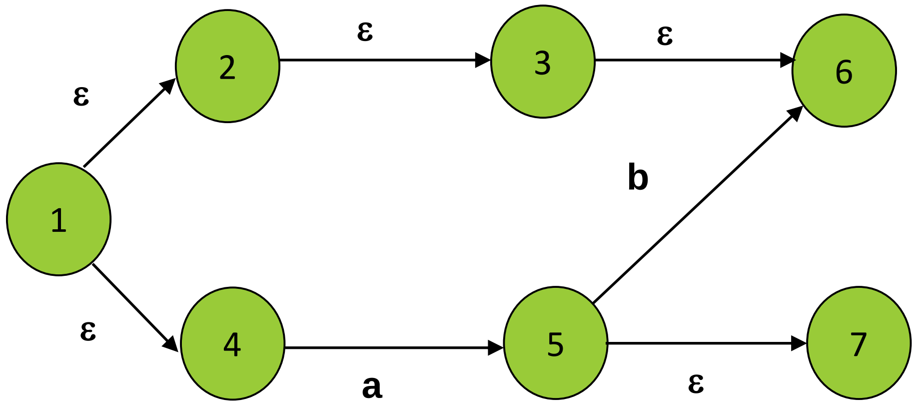
Induction: If p is in state ECLOSE(q) and there is a transition form p to r using **ε,** then *r is in ECLOSE(q).*

$ECLOSE(q_0)=\{q_0 , q_1\}$

$ECLOSE(q_3)=\{q_3 , q_5\}$



Epsilon-Closures



ECLOSE(1)

={1,2,3,4,6}

ECLOSE(2)

={2,3,6}

Extended Transitions & Languages for ε-NFA's

– Recursive definition of extended transition function $\hat{\delta}$:

Basis: $\hat{\delta}(q, \varepsilon) = \text{ECLOSE}(q)$.

Induction: if $w = xa$, then $\hat{\delta}(q, w)$ is computed as:

If $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_k\}$ and

$$\delta(p_i, a) = \{r_1, r_2, \ldots, r_m\},$$

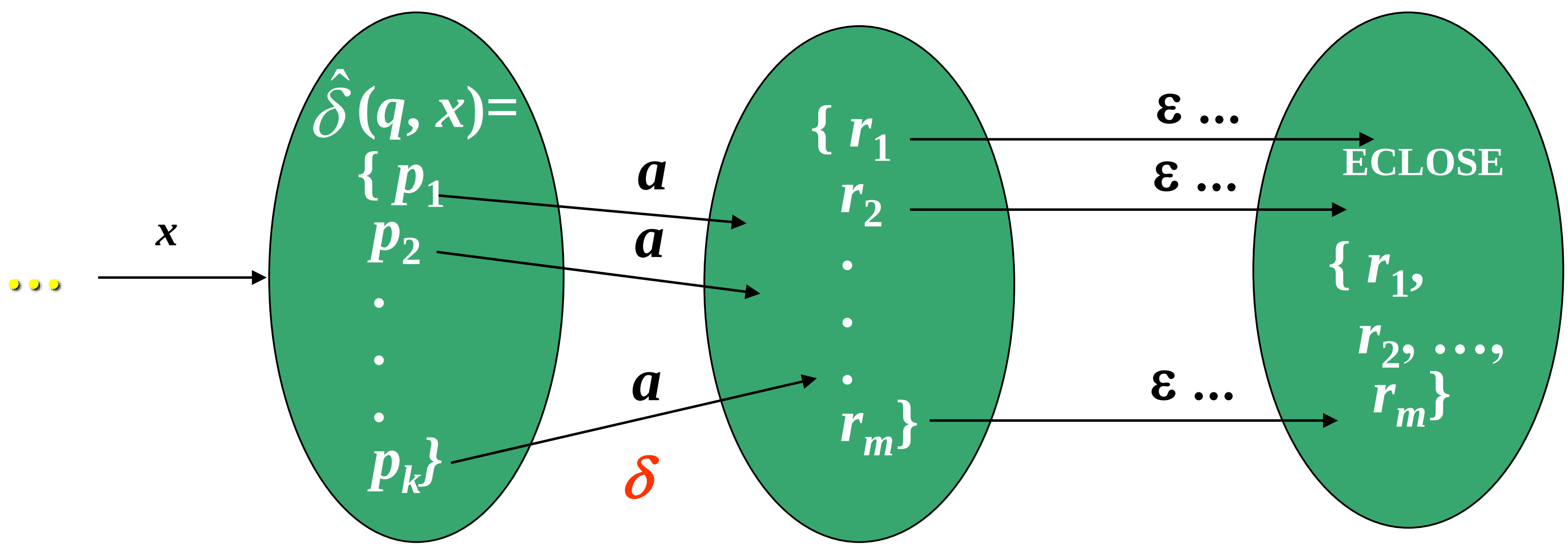
then $\hat{\delta}(q, w) = \text{ECLOSE}(\{r_1, r_2, \ldots, r_m\}) = \text{ECLOSE}(\bigcup_{i=1}^{k} \delta(p_i, a))$.

Extended Transitions & Languages for ε-NFA's..

Induction: if $w = xa$, then (q, w) is computed as:

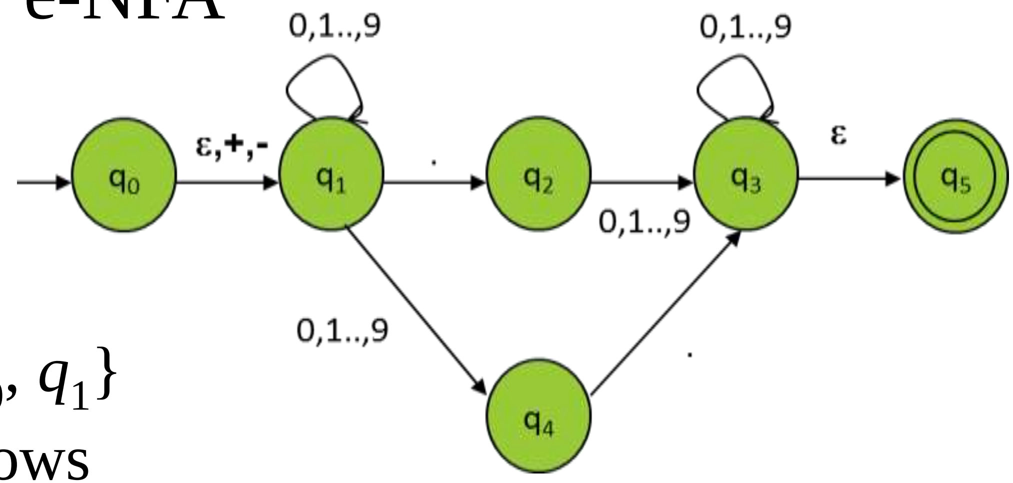
If $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_k\}$ and $\hat{\delta}(p_i, a) = \{r_1, r_2, \ldots, r_m\}$,

then $\hat{\delta}(q, w) = \text{ECLOSE}(\{r_1, r_2, \ldots, r_m\})$.



FA with ε-transition

Computing $\hat{\delta}$ $(q_0, 5.6)$ for e-NFA



- $\hat{\delta}$ $(\boldsymbol{q_0}, e)$ =ECLOSE(q_0)=$\{q_0, q_1\}$
- Compute $\hat{\delta}$ $(q_0, 5)$ as follows

1. $\hat{\delta}$ $(q_0, 5)$ = $(q_0, e5)$ = ECLOSE$(\delta(q_0, 5) \cup \delta(q_1, 5))$=$\{q_1, q_4\}$
2. ECLOSE the result of step (1)

$$= \text{ECLOSE}(\{q_1, q_4\}) = \text{ECLOSE}(\{q_1\}) \cup \text{ECLOSE}(\{q_4\})$$
$$= \{q_1, q_4\}$$

- Compute $\hat{\delta}$ $(q_0, 5.)$
- Compute $\hat{\delta}$ $(q_0, 5.6)$

Eliminating ε-Transitions

Eliminating ε-Transitions

- The ε-transition is good for design of FA, but for implementation, they have to be eliminated.
- Given an ε-NFA, we can find an equivalent DFA (a theorem seen later).
- Let $\boldsymbol{E = (Q_E, S, \delta_E, q_0, F_E)}$ be the given ε-NFA, the equivalent DFA

  $\boldsymbol{D = (Q_D, S, \delta_D, q_D, F_D)}$ is constructed

Eliminating ε-Transitions..

- Q_D is the set of subsets of Q_E, in which each accessible is an e-closed subset of Q_E, i.e., are sets $S \subseteq Q_E$ such that S = ECLOSE(S).

  In other words, each e-closed set of states, S, includes those states such that any e-transition out of one of the states in S leads to a state that is also in S.
- q_D = ECLOSE(q_0) (initial state of D)
- $F_D = \{S \mid S \in Q_D \text{ and } S \cap F_E \neq \phi\}$

Eliminating ε-Transitions..

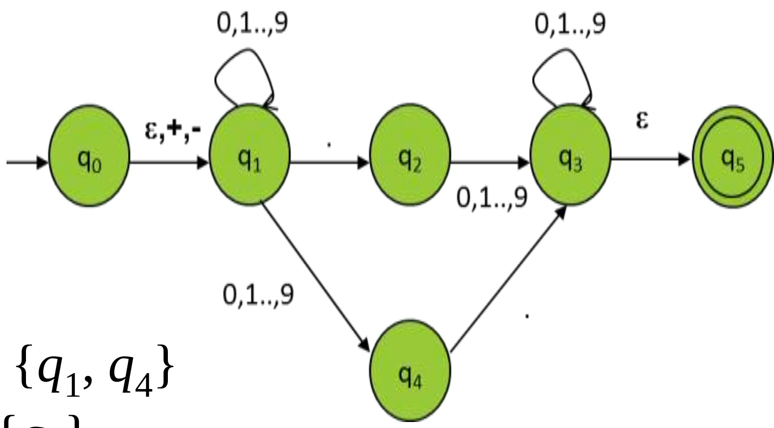
- $\delta_D(S, a)$ is computed for each a in Σ and each S in Q_D in the following way:
  - Let $S = \{p_1, p_2, ..., p_k\}$
  - Compute $\bigcup_{i=1}^{k} \delta(p_i, a)$ and let this set be $\{r_1, r_2, \ldots, r_m\}$
  - Set $\delta_D(S, a) = \text{ECLOSE}(\{r_1, r_2, \ldots, r_m\})$

    $= \text{ECLOSE}(\bigcup_{j=1}^{m} ECLOSE(rj))$

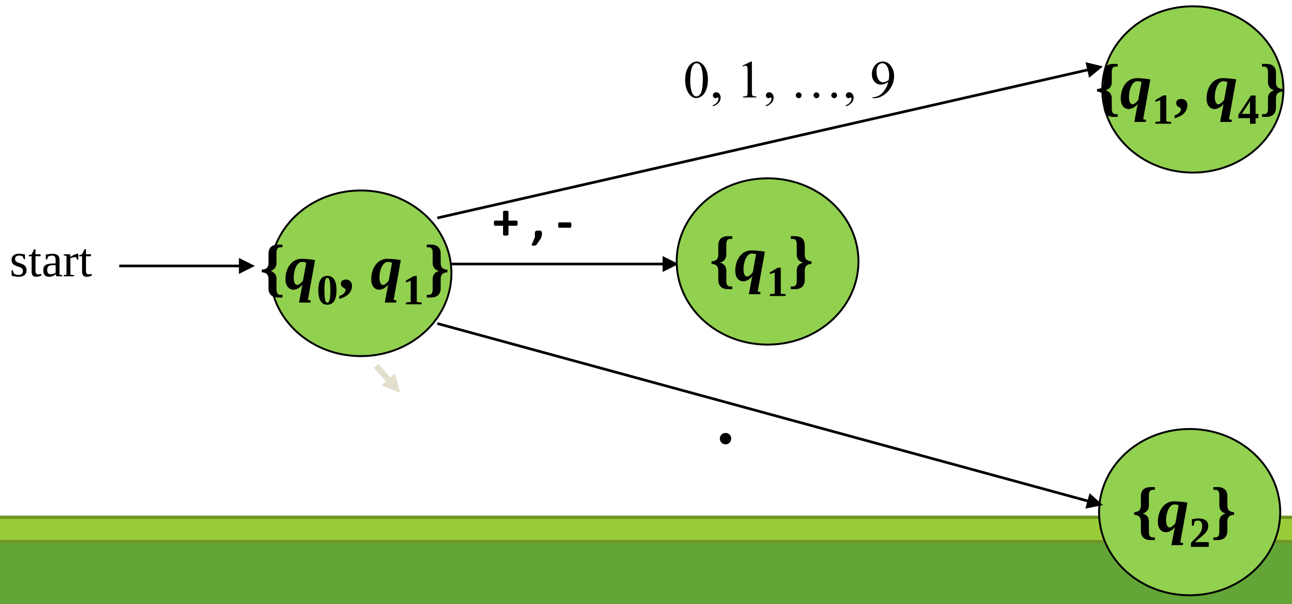
– Technique to create accessible states in DFA D:
  - starting from the start state q_0 of ε-NFA E, generate ECLOSE(q_0) as start state q_D of D;
  - from the generated states to derive other states.

Finite Automata with Epsilon-Transitions

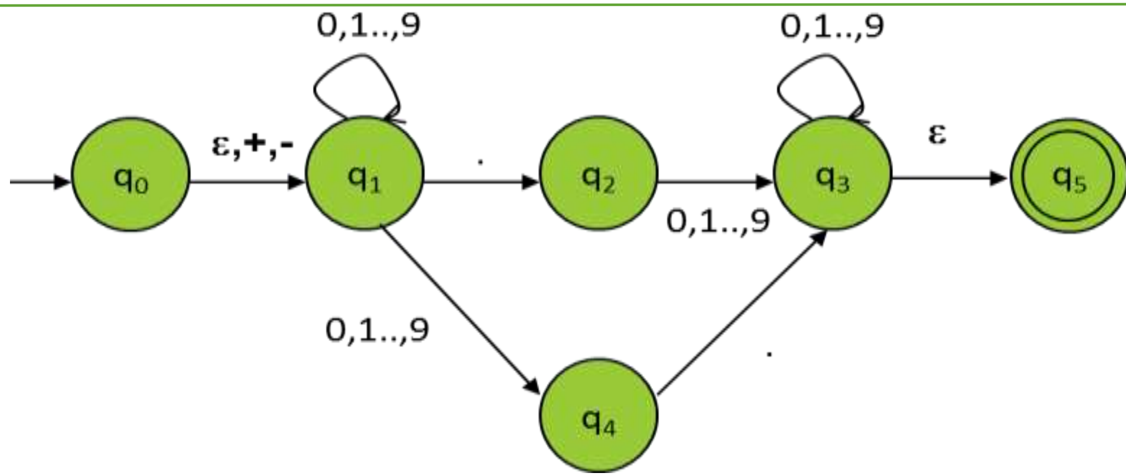
- Start state q_D = ECLOSE(q_0) = $\{q_0, q_1\}$
- $d_D(\{q_0, q_1\}, +)$ = ECLOSE($d_E(q_0, +) \cup d_E(q_1, +)$)
  = ECLOSE($\{q_1\} \cup \phi$) = ECLOSE($\{\mathbf{q_1}\}$) = $\{q_1\}$



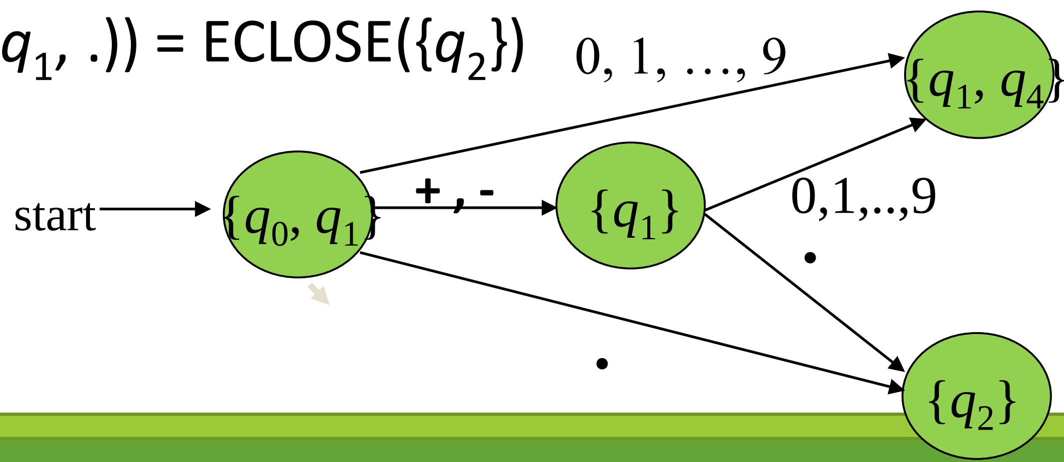
- $d_D(\{q_0, q_1\}, 0)$ = ECLOSE($d_E(q_0, 0) \cup d_E(q_1, 0)$)
  = ECLOSE ($\phi \cup \{q_1, q_4\}$) = ECLOSE($\{q_1, q_4\}$) = $\{q_1, q_4\}$
- $d_D(\{q_0, q_1\}, .)$ = ECLOSE($d_E(q_0, .) \cup d_E(q_1, .)$) = $\{q_2\}$



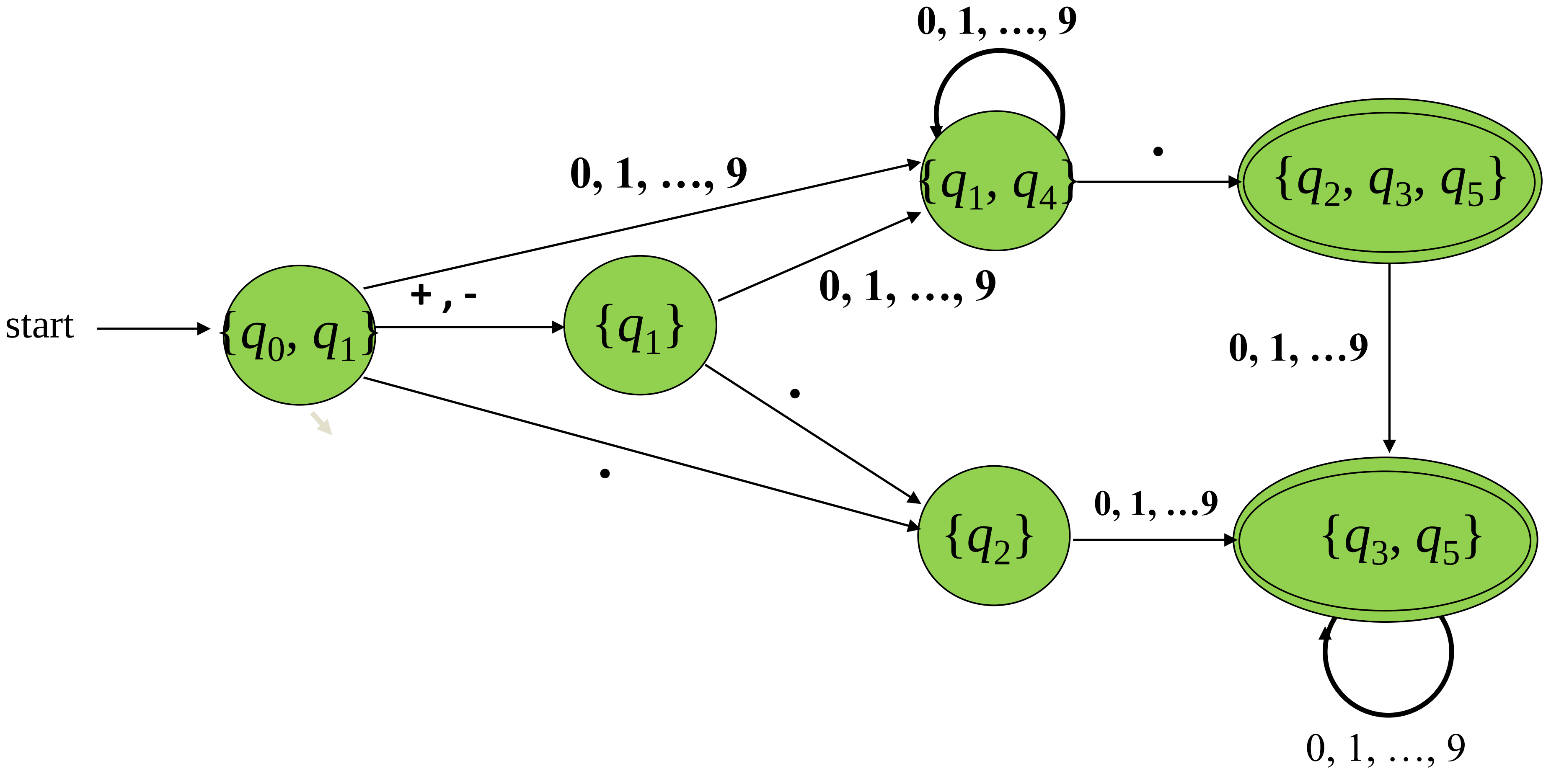
Finite Automata with Epsilon-Transitions



- $\delta_D(\{q_1\}, 0) = \text{ECLOSE}(\delta_E(q_1, 0)) = \text{ECLOSE}(\{q_1, q_4\})$
  $= \{q_1, q_4\}$...
- $\delta_D(\{q_1\}, .) = \text{ECLOSE}(\delta_E(q_1, .)) = \text{ECLOSE}(\{q_2\})$
  $= \{q_2\}$

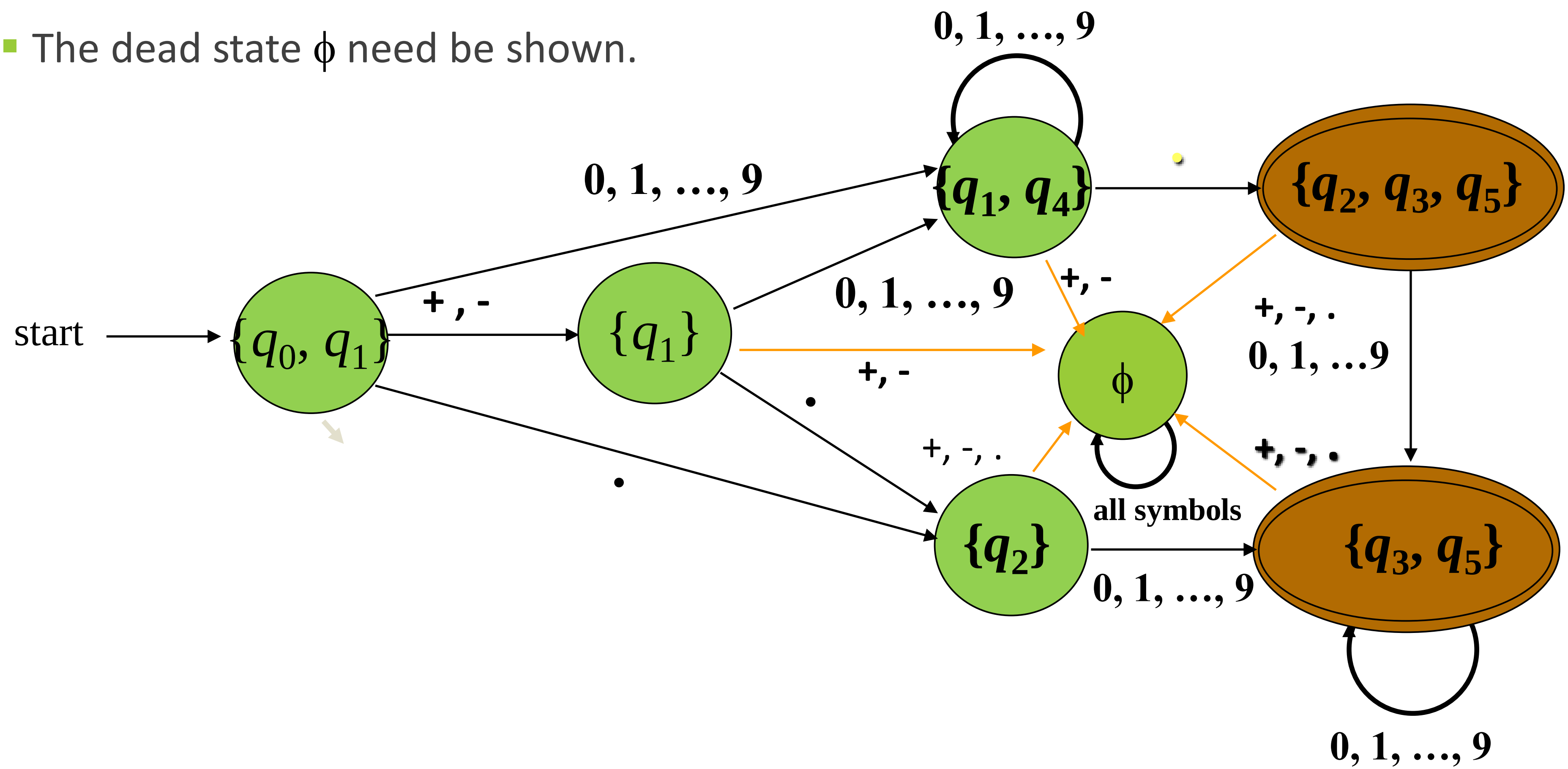


Finite Automata with Epsilon-Transitions

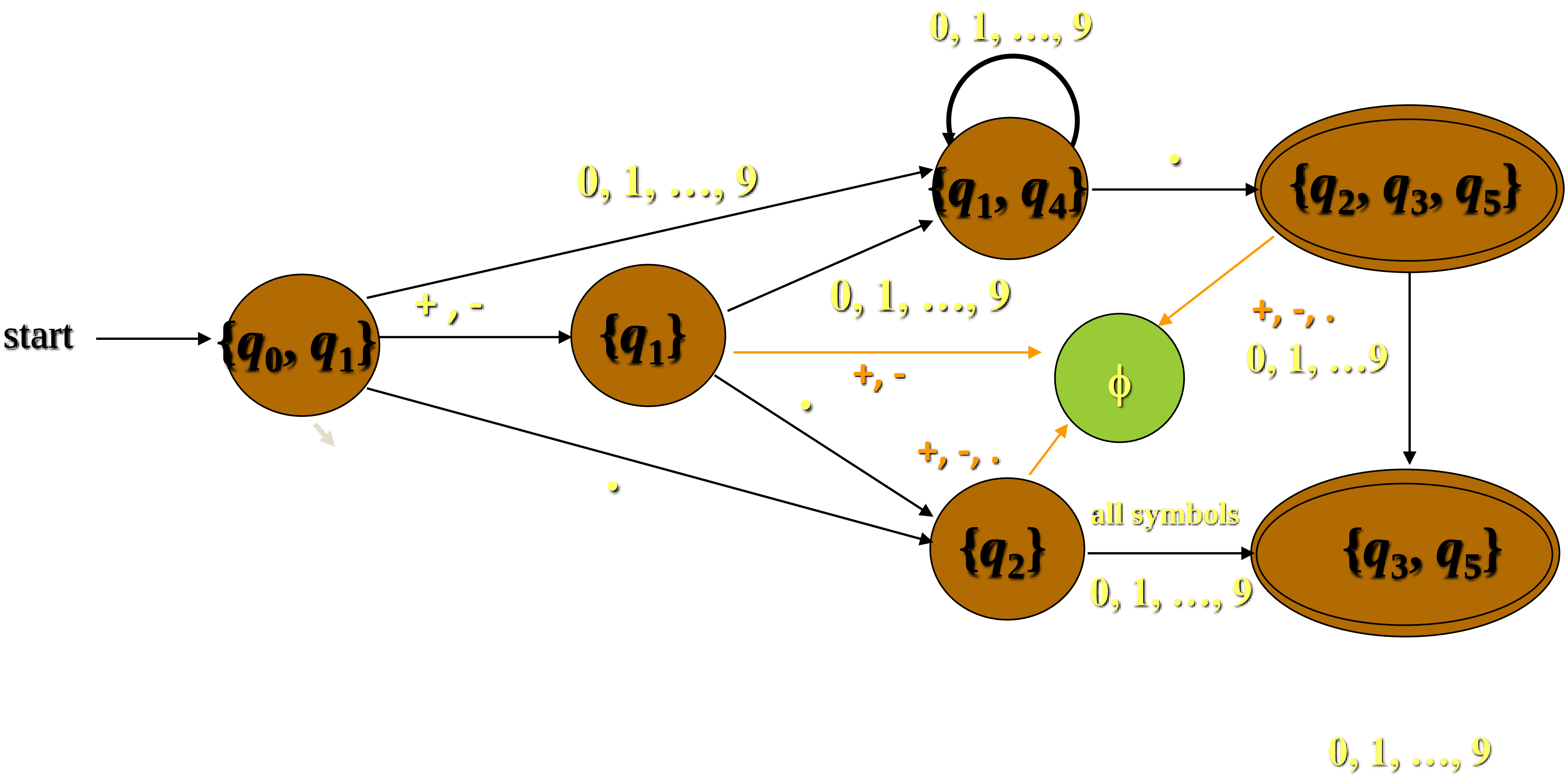


Finite Automata with Epsilon-Transitions

- The dead state ϕ need be shown.



Finite Automata with Epsilon-Transitions



Thank you ☺