Chapter 2

Finite Automata

(part B)
(2015/10/02)

Windmills in Holland

Outline

2.4 An Application — Text Search

2.4.1 Finding Strings in Text

- Concept: searching Google for a set of words is equivalent to just finding strings in documents
- Techniques
  - ♦ Using inverted indexes
  - ♦ Using finite automata
- Technique using inverted indexes --- as illustrated by Fig. 2.14.

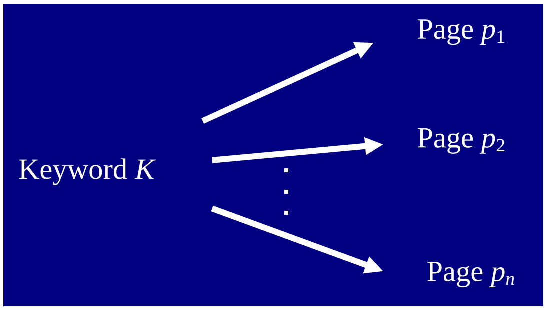

Fig. 2.14 An illustration of the technique of inverted indexes.

- Applications unsuitable to use inverted indexing:
  - ♦ The document repository changes rapidly.
  - ♦ Documents to be searched cannot be catalogued.

2.4.2 NFA's for Text Search

- A simple example (supplemental) ---

Find the DFA equivalent to the NFA which searches words of $x_1 = ab$ and $x_2 = b$ in long text strings over the alphabet $\Sigma = \{a, b\}$.

- ♦ Such words y_1 and y_2 may be described as

  Segment $x_1 = ab$ or $x_2 = b$ following any string of a and b, and followed by any string of a and b.

- ♦ Equivalently, the two words are two strings y_1 and y_2 which may be described by concatenations as

$$y_1 = z_1x_1z_2$$
$$y_2 = z_3x_2z_4$$

  where z_i with $i = 1, 2, 3, 4$ is any string of a and b
- ♦ An intuitive solution in NFA form according to the above discussion can be drawn directly as shown in Fig. 2.15.

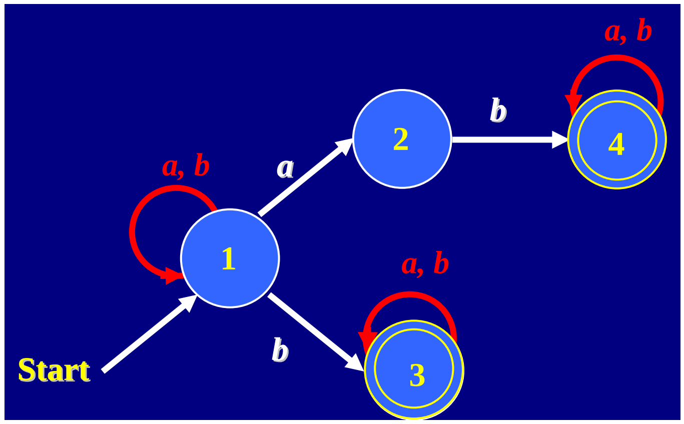

Fig. 2.15 An NFA for searching words of $x_1 = ab$ and $x_2 = b$ in long text strings.

♦ By the lazy evaluation method, the transition table of a DFA equivalent to the NFA in Fig. 15 is shown in Table 2.6 (note: state 12 means state {1, 2}, etc.). The detail of obtaining the DFA is left as an exercise.

Table 2.6 The transition table for the DFA which is equivalent to the NFA of Fig. 2.15.

	a	*b*
→{1}	{1,2}	{1, 3}
{1, 2}	{1, 2}	{1, 3, 4}
*{1, 3}	{1, 2, 3}	{1, 3}
*{1, 2, 3}	{1, 2, 3}	{1, 3, 4}
*{1, 3, 4}	{1, 2, 3, 4}	{1, 3, 4}
*{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 3, 4}

♦ The transition diagram of the DFA equivalent to the NFA is shown in Fig. 2.16.

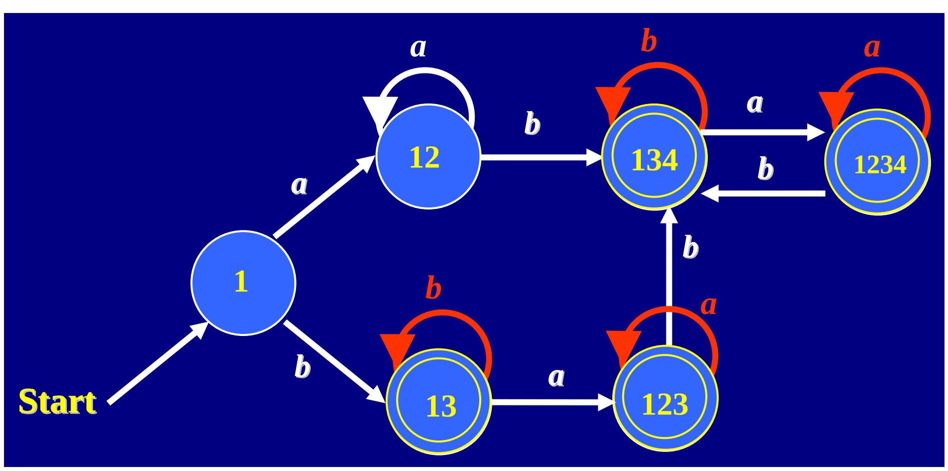

Fig. 2.16 A DFA equivalent to the NFA in Fig. 2.15.

2.4.3 A DFA to Recognize a Set of Keywords

- Example 2.14 ---

  use an NFA to search two keywords "web" and "eBay" among text.

  - Derive the desired NFA in a way as described previously, which is shown in Fig. 2.17.

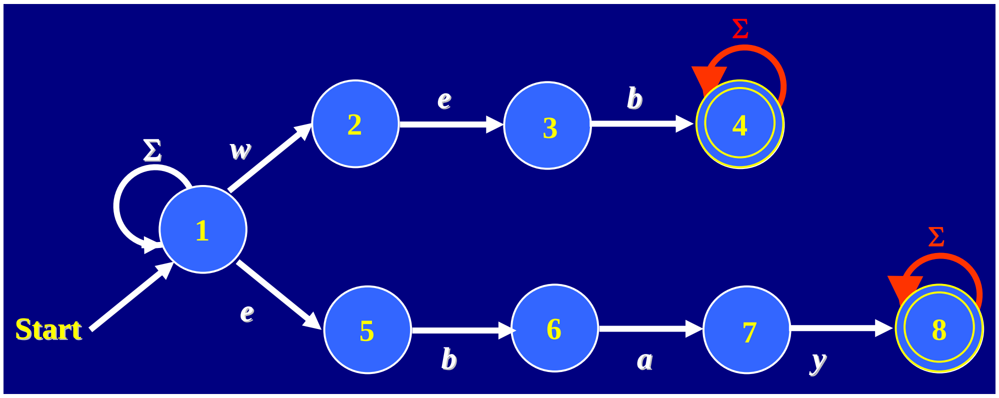

Fig. 2.17 The NFA for Example 2.14.

  - How to implement the NFA ?
    - Write a *simulation program* like Fig. 2.8 which is repeated here as Fig. 2.18 below.

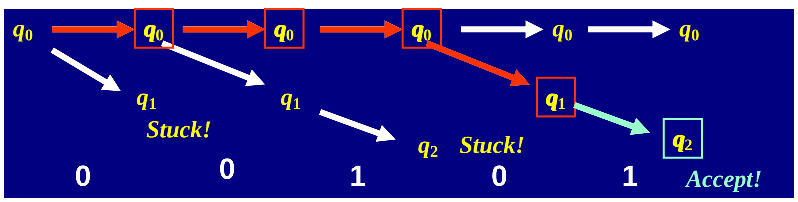

Fig. 2.18 The NFA of Example 2.2 accepting x = 00101 which can be implemented by a simulation program.

    - Convert the NFA to an equivalent DFA using subset construction and simulate the DFA, which is shown as the Fig. 17 in the textbook (*not* the Fig. 17 above in this lecture note!!!).
    - Some rules may be inferred *as a theorem* for constructing *directly* this kind of keyword-recognition DFA. (For the details, read the textbook by yourself.)

2.5 Finite Automata with Epsilon-Transitions

2.5.1 Use of ε-transitions

- Concepts ---
  - We allow the automaton to accept the empty string ε.
  - This means that a transition is allowed to occur *without* reading in a symbol.
  - The resulting NFA is called ε-NFA.
  - It adds "programming (design) convenience" (more intuitive for use in designing FA's)

■ Example 2.16 ---

An ε-NFA accepting decimal numbers like 2.15, .125, +1.4, -0.501… is as shown in Fig. 2.19.

♦ To accept a number like "+5." (nothing after the decimal point), we have to add q_4.

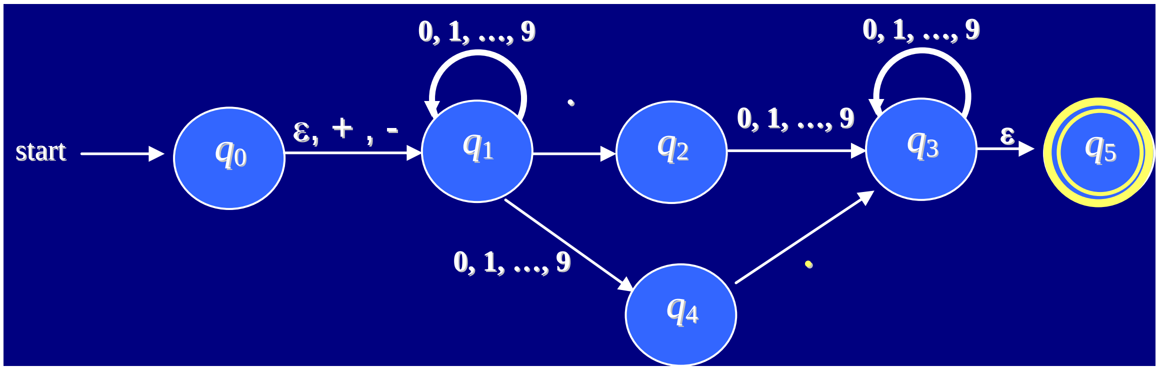

Fig.2.19 An ε-NFA accepting decimal numbers.

■ Example 2.17 ---

A more intuitive ε-NFA for Example 2.14 is shown in Fig. 2.20.

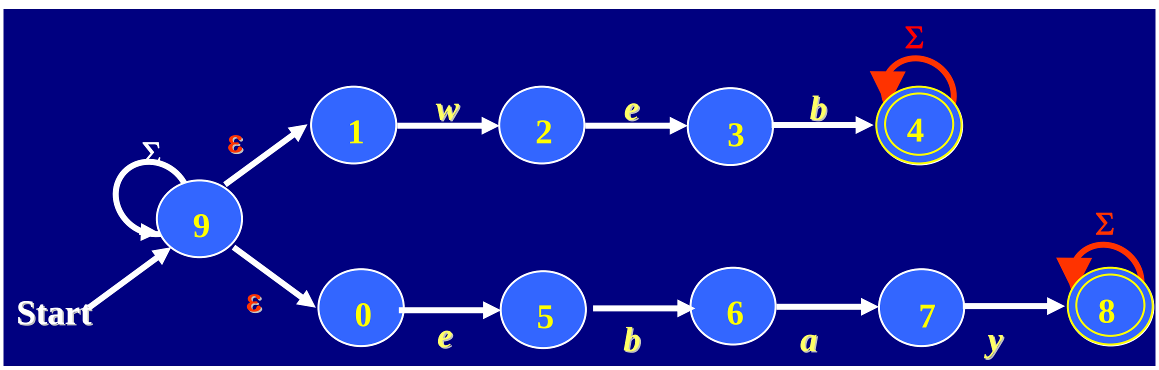

Fig. 2.20 A more intuitive ε-NFA for Example 2.14.

2.5.2 <u>Formal Notation for an ε-NFA</u>

■ Definition ---

An ε-NFA A is denoted by $A = (Q, \Sigma, \delta, q_0, F)$ where the transition function δ takes the following as arguments:

♦ a state in Q, and

♦ a member of $\Sigma \cup \{\varepsilon\}$.

■ Notes ---

♦ The empty string ε cannot be used as an input symbol, but can be accepted to yield a transition!

♦ $\delta(q_i, \varepsilon)$ is defined for every state q_i (getting into the dead state ϕ if there is no next state with ε as input symbol originally).

■ Example 2.18 ---

The ε-NFA of Fig. 2.19 is described as follows.

♦ $E = (\{q_0, q_1, ..., q_5\}, \{., +, -, 0, 1, ..., 9\}, \delta, q_0, \{q_5\})$.

♦ The transitions include, e.g.,
- $\delta(q_0, \varepsilon) = \{q_1\}$;
- $\delta(q_1, \varepsilon) = \phi$.

♦ See Table 2.7 for a complete transition table of E.

Table 2.7 The transition table for the ε-NFA of example 2.18.

	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

2.5.3 Epsilon-Closures (ε-closures)

■ Concepts ---

♦ We have to define the ε-closure to define the extended transition function for the ε-NFA.

♦ We "ε-closure" a state q by following all transitions out of q that are labeled ε.

■ Formal recursive definition of *the set* ECLOSE(q) for q ---

♦ *Basis*: state q is in ECLOSE(q) (i.e., ECLOSE(q) includes the state q itself);

♦ *Induction*: if p is in ECLOSE(q), then all states accessible from p through paths of ε*'s* are also in ECLOSE(q).
(A metaphor: those "state stations" accessible through "ε-type highways" are all included in ECLOSE(q).)
(比喻："經ε*式快速道路路段*可到的所有 *states 車站*皆算在 ECLOSE(q)之內")

■ Definition of the ε-closure of a set S of states ---

$$\text{ECLOSE}(S) = \bigcup_{q \in S} \text{ECLOSE}(q).$$

■ Example 2.19 ---

Given an ε-NFA as shown in Fig. 2.21, we have

♦ ECLOSE(1) = {1, 2, 3, 4, 6};

♦ ECLOSE({3, 5}) = ECLOSE(3) ∪ ECLOSE(5) = {3, 6} ∪ {5, 7} = {3, 5, 6, 7}.

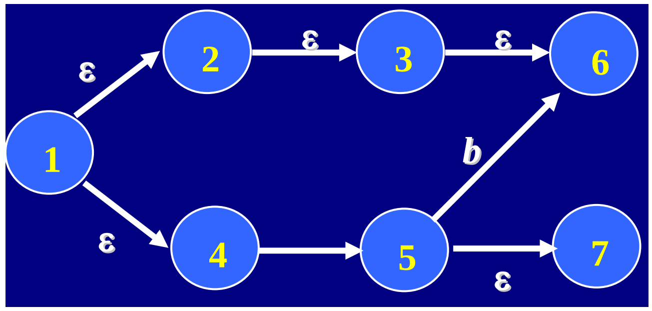

Fig. 2.21 The ε-NFA of Example 2.19.

2.5.4 <u>Extended Transitions & Languages for ε-NFA's</u>

- Recursive definition of extended transition function $\hat{\delta}$ ---
  - *Basis*: $\hat{\delta}(q, \varepsilon) = \mathrm{ECLOSE}(q)$.
  - *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 $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$, then

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

- An illustration of the above definition is shown in Fig. 2.22.

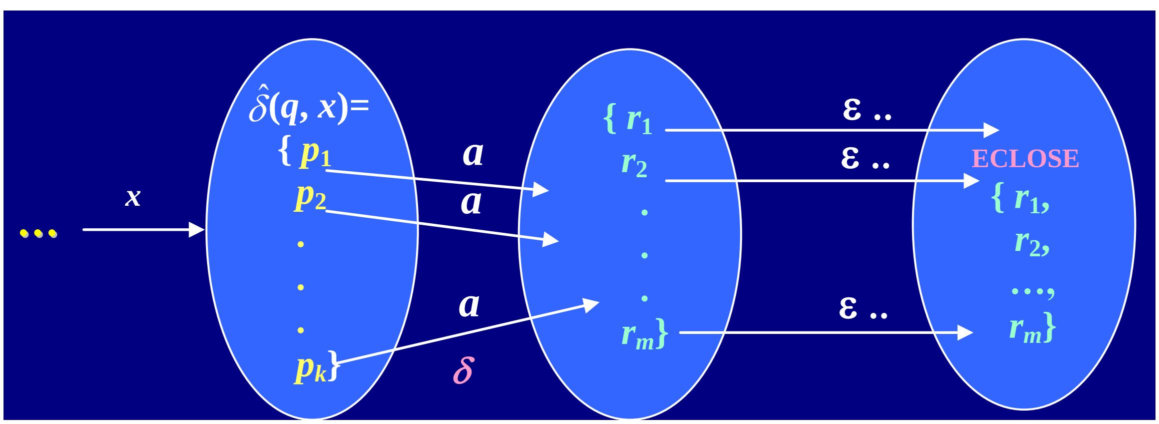

Fig. 2.22 A illustration of computing the recursive definition of extended transition function $\hat{\delta}$.

- Example 2.20 ---

Compute $\hat{\delta}(q_0, 5.6)$ for the ε-NFA of Fig. 2.19.

- Compute $\hat{\delta}(q_0, 5) = \hat{\delta}(q_0, \varepsilon 5)$ first:
  - $\hat{\delta}(q_0, \varepsilon) = \mathrm{ECLOSE}(q_0) = \{q_0, q_1\}$;
  - $\delta(q_0, 5) \cup \delta(q_1, 5) = \phi \cup \{q_1, q_4\} = \{q_1, q_4\}$;
  - $\hat{\delta}(q_0, 5) = \hat{\delta}(q_0, \varepsilon 5) = \mathrm{ECLOSE}(\delta(q_0, 5) \cup \delta(q_1, 5))$
    $= \mathrm{ECLOSE}(\{q_1, q_4\}) = \mathrm{ECLOSE}(\{q_1\}) \cup \mathrm{ECLOSE}(\{q_4\})$

$= \{q_1, q_4\}$.

(Complete the rest of computations by yourself!)

- The language of an ε-NFA ---

Given an ε-NFA $E = (Q, \Sigma, \delta, q_0, F)$, the language accepted by it is

$$L(E) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \phi\}.$$

2.5.5 Eliminating ε-Transitions

- Concepts ---
  - 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).

- *Proof of equivalence* of the ε-NFA and the DFA ---
  - Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be the given ε-NFA, the equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ is constructed as follows.
    - Q_D is the set of subsets of Q_E in which each subset S is accessible as an ε-closed subset of Q_E, i.e., $S \subseteq Q_E$ such that S = ECLOSE(S).

    (In other words, each ε-closed set S of states includes those states such that any ε-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).
    - $F_D = \{S \mid S \in Q_D \text{ and } S \cap F_E \neq \phi\}$.
    - $\delta_D(S, a)$ is computed for each a in Σ and each S in Q_D in the following way:

      (1) let $S = \{p_1, p_2, ..., p_k\}$;

      (2) compute $\bigcup_{i=1}^{k} \delta(p_i, a)$ and let this set be $\{r_1, r_2, \ldots, r_m\}$;

      (3) set $\delta_D(S, a) = \text{ECLOSE}(\{r_1, r_2, \ldots, r_m\}) = \text{ECLOSE}(\bigcup_{i=1}^{k} \delta(p_i, a))$.

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

- Example 2.21 ---

Eliminate the ε-transitions of Fig. 2.19 above.

  - Start state $q_D = \text{ECLOSE}(q_0) = \{q_0, q_1\}$.
  - $\delta_D(\{q_0, q_1\}, +) = \text{ECLOSE}(\delta_E(q_0, +) \cup \delta_E(q_1, +))$.

    $= \text{ECLOSE}(\{q_1\} \cup \phi) = \text{ECLOSE}(\{q_1\}) = \{q_1\}$, ...
  - $\delta_D(\{q_0, q_1\}, 0) = \text{ECLOSE}(\delta_E(q_0, 0) \cup \delta_E(q_1, 0))$

    $= \text{ECLOSE}(\phi \cup \{q_1, q_4\}) = \text{ECLOSE}(\{q_1, q_4\}) = \{q_1, q_4\}$, ...
  - $\delta_D(\{q_0, q_1\}, .) = \text{ECLOSE}(\delta_E(q_0, .) \cup \delta_E(q_1, .)) = \{q_2\}$.

(The above partial derivation may be described by Fig. 2.23.)

- $_D(\{q_1\}, 0)$ = ECLOSE($_E(q_1, 0)$) = ECLOSE($\{q_1, q_4\}$) = $\{q_1, q_4\}$…
- $_D(\{q_1\}, .)$ = ECLOSE($_E(q_1, .)$) = ECLOSE($\{q_2\}$) = $\{q_2\}$.

(The above partial derivation may be described by Fig. 2.24.)

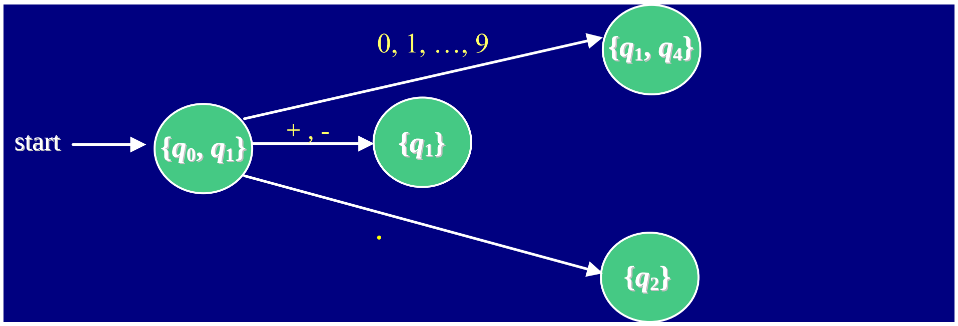

Fig. 2.23 The first partial derivation of the equivalent DFA of the ε-NFA of Example 2.21.

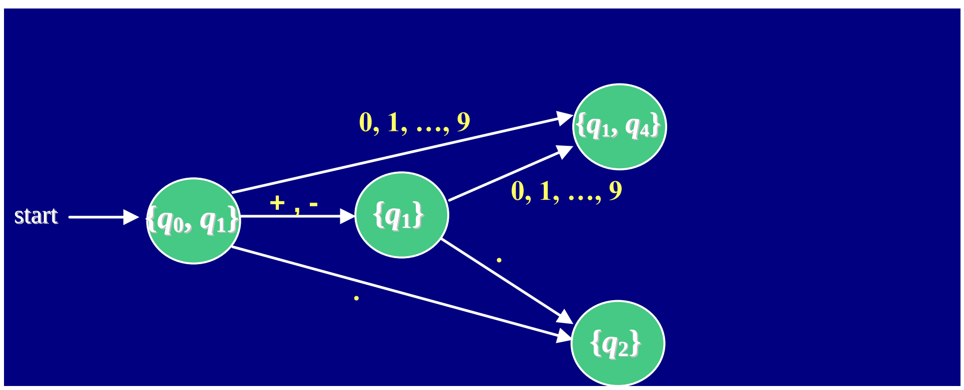

Fig. 2.24 The second partial derivation of the equivalent DFA of the ε-NFA of Example 2.21.

- $\delta_D(\{q_1, q_4\}, 0) = \text{ECLOSE}(\delta_E(q_1, 0) \cup \delta_E(q_4, 0)) = \text{ECLOSE}(\{q_1, q_4\} \cup \phi) = \{q_1, q_4\}$…
- $\delta_D(\{q_1, q_4\}, .) = \text{ECLOSE}(\delta_E(q_1, .) \cup \delta_E(q_4, .)) = \text{ECLOSE}(\{q_2\} \cup \{q_3\})$
  $= \text{ECLOSE}(q_2) \cup \text{ECLOSE}\ (q_3) = \{q_2\} \cup \{q_3, q_5\} = \{q_2, q_3, q_5\}$.

(The above partial derivation may be described by Fig. 2.25.)

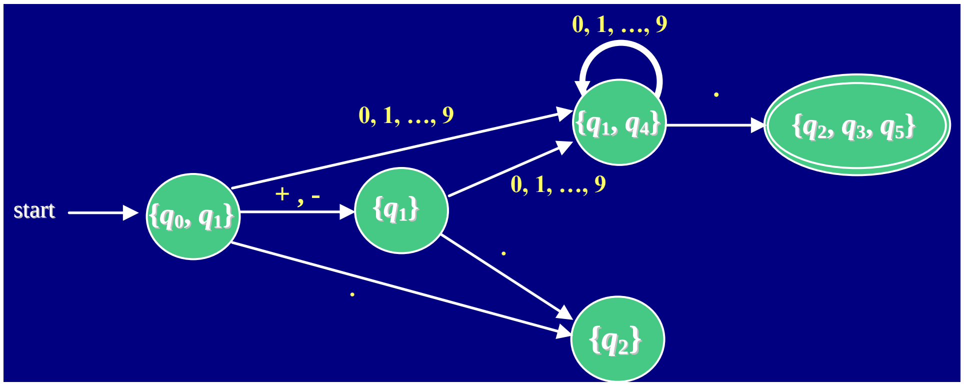

Fig. 2.25 The third partial derivation of the equivalent DFA of the ε-NFA of Example 2.21.

- ♦ $\delta_D(\{q_2\}, 0) = \text{ECLOSE}(\delta_E(q_2, 0)) = \text{ECLOSE}(\{q_3\}) = \{q_3, q_5\}\ldots$
- ♦ $\delta_D(\{q_2, q_3, q_5\}, 0) = \text{ECLOSE}(\delta_E(q_2, 0) \cup \delta_E(q_3, 0) \cup \delta_E(q_5, 0))$
  $= \text{ECLOSE}(\{q_3\} \cup \{q_3\} \cup \phi) = \text{ECLOSE}(q_3) = \{q_3, q_5\}\ldots$

(The above partial derivation may be described by Fig. 2.26.)

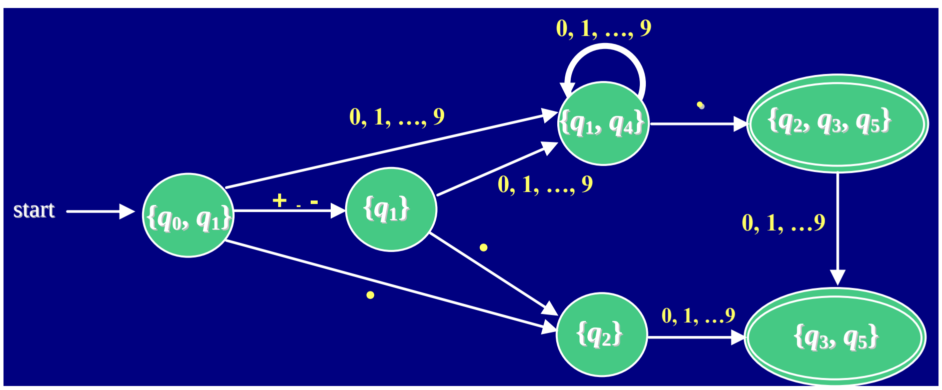

Fig. 2.26 The fourth partial derivation of the equivalent DFA of the ε-NFA of Example 2.21.

- ♦ $\delta_D(\{q_3, q_5\}, 0) = \text{ECLOSE}(\delta_E(q_3, 0) \cup \delta_E(q_5, 0)) = \text{ECLOSE}(\{q_3\} \cup \phi) = \text{ECLOSE}(q_3)$
  $= \{q_3, q_5\}\ldots$

(The above fifth derivation may be described by Fig. 2.27.)

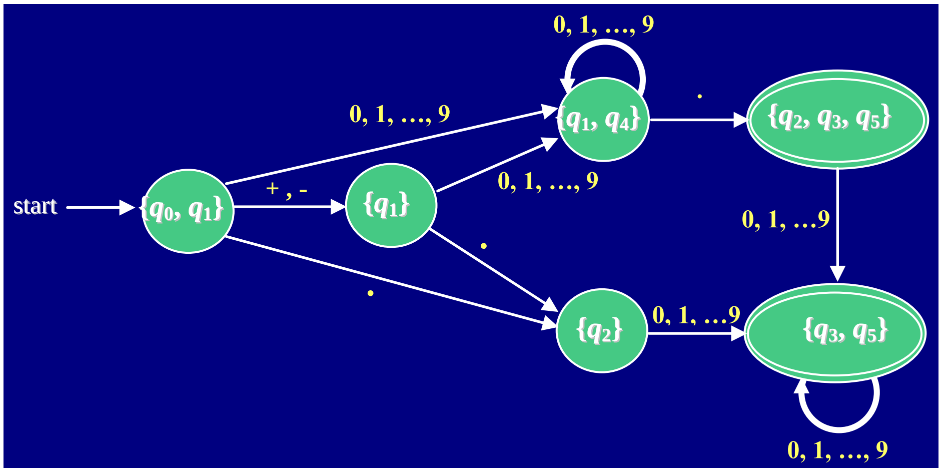

Fig. 2.27 The fifth partial derivation of the equivalent DFA of the ε-NFA of Example 2.21.

- ♦ The dead state ϕ need be shown to get the final version of the desired DFA as shown in Fig. 28.
- ♦ But according to Section 2.3.6, the diagram in Fig. 27 may be regarded as *deterministic*.

(A suggestion to the reader: you would better repeat the above entire derivation process to get a full understanding of the involved details.)

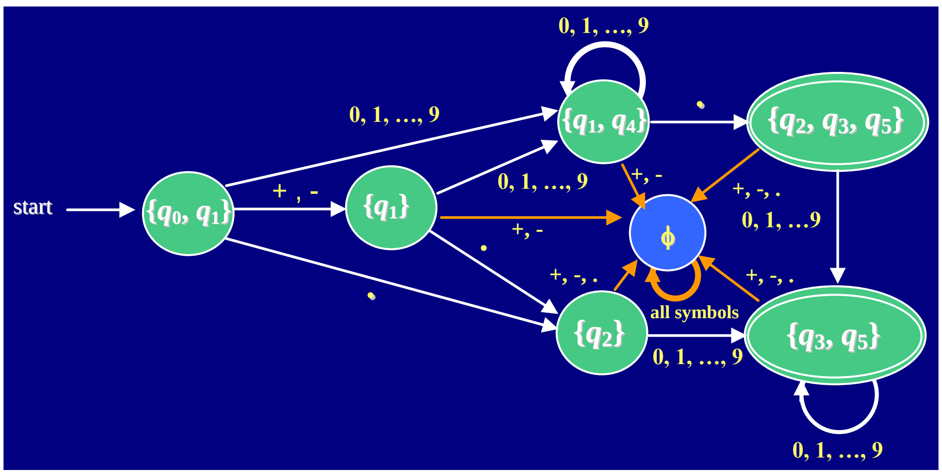

Fig. 2.28 The complete derivation of the equivalent DFA of the ε-NFA of Example 2.21.

■ Theorem 2.22 ---

A language L accepted by some ε-NFA if and only if L is accepted by some DFA.

- ♦ Proof: see the textbook yourself.

■ A Review ---

- ♦ 3 Types of Automata:
  - • DFA --- good for soft/hardware implementation;
  - • NFA ($\not\varepsilon$-NFA) --- intermediately intuitive;

• ε-NFA --- most intuitive

(Note: notation $\not\varepsilon$ in the above is pronounced as “non-epsilon.”)

♦ Some equivalence relations among these 3 types of automata are shown in Fig. 2.29.

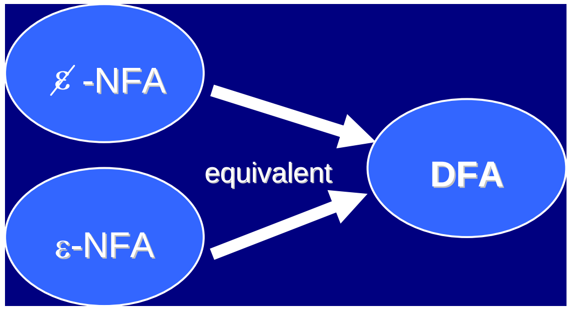

Fig. 2.28 Some relations among the three types of automata DFA, $\not\varepsilon$ -NFA, and ε-NFA.