CSE-217: Theory of Computation

NON-DETERMINISM

Md Jakaria

Lecturer
Department of Computer Science and Engineering
Military Institute of Science and Technology

August 7, 2019

FORMAL DEFINITION

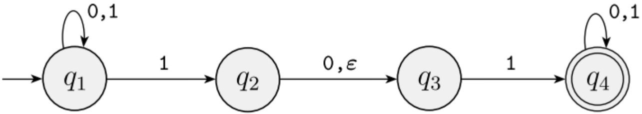
NON-DETERMINISM

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.

NON-DETERMINISM



NON-DETERMINISM

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. $F = \{q_4\}$.

Equivalence of NFA and DFA

1. Deterministic and nondeterministic finite automata recognize the same class of languages.
2. Such equivalence is both surprising and useful.

Equivalence of NFA and DFA

3 It is surprising because NFAs appear to have more power than DFAs, so we might expect that NFAs recognize more languages.

4 It is useful because describing an NFA for a given language sometimes is much easier than describing a DFA for that language.

5 Say that two machines are equivalent if they recognize the same language.

Equivalence of NFA and DFA

Theorem

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

Equivalence of NFA and DFA

PROOF IDEA

1. If a language is recognized by an NFA, then we must show the existence of a DFA that also recognizes it.
2. The idea is to convert the NFA into an equivalent DFA that simulates the NFA.
3. Recall the “reader as automaton” strategy for designing finite automata.

Equivalence of NFA and DFA

4 How would you simulate the NFA if you were pretending to be a DFA?

5 What do you need to keep track of as the input string is processed?

6 In the examples of NFA's, you kept track of the various branches of the computation by placing a finger on each state that could be active at given points in the input.

Equivalence of NFA and DFA

7. You updated the simulation by moving, adding, and removing fingers according to the way the NFA operates.
8. All you needed to keep track of was the set of states having fingers on them.
9. If k is the number of states of the NFA, it has 2^k subsets of states.

Equivalence of NFA and DFA

10 Now we need to figure out which will be the start state and accept states of the DFA.

11 What will be its transition function.

12 We can discuss this more easily after setting up some formal notation.

Equivalence of NFA and DFA

PROOF

- Let N = (Q, Σ, δ, q_0, F) be the NFA recognizing some language A
- We construct a DFA M = (Q', Σ', δ', q_0', F') recognizing A

Equivalence of NFA and DFA

- Before doing the full construction, lets first consider the easier case wherein N has no ϵ arrows.
- Later we take the ϵ arrows into account.

Equivalence of NFA and DFA

1. $Q' = P(Q)$.
   - Every state of M is a set of states of N.
   - Recall that P(Q) is the set of subsets of Q.

Equivalence of NFA and DFA

2 For $R \in Q'$ and $a \in \Sigma$, let

$$\delta'(R, a) = \{q \in Q | q \in \delta(r, a) \text{ for some } r \in R\}$$

- If R is a state of M, it is also a set of states of N.
- When M reads a symbol a in state R, it shows where a takes each state in R.
- Because each state may go to a set of states, we take the union of all these sets

Equivalence of NFA and DFA

3 $q'_0 = \{q_0\}$

- M starts in the state corresponding to the collection containing just the start state of N.

Equivalence of NFA and DFA

4. $F' = \{R \in Q' | R$ contains an accept state of N$\}$
   - The machine M accepts if one of the possible states that N could be in at this point is an accept state.

Equivalence of NFA and DFA

- Now we need to consider the ϵ arrows.
- To do so, we set up an extra bit of notation.
- For any state R of M, we define E(R) to be the collection of states that can be reached from members of R by going only along ϵ arrows, including the members of R themselves

Equivalence of NFA and DFA

- Formally, for $R \subseteq Q$ let $E(R) = \{q|q$ can be reached from R by traveling along 0 or more ϵ arrows$\}$
- Then we modify the transition function of M to place additional fingers on all states that can be reached by going along ϵ arrows after every step.
- Replacing $\delta(r, a)$ by $E(\delta(r, a))$ achieves this effect

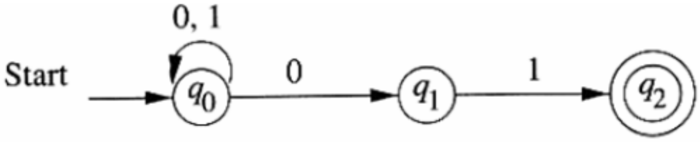
Equivalence of NFA and DFA

- Thus $\delta'(R, a) = \{q \in Q | q \in E(\delta(r, a))$ for some $r \in R\}$
- Additionally, we need to modify the start state of M to move the fingers initially to all possible states that can be reached from the start state of N along the ϵ arrows.
- Changing q_0' to be $E(\{q_0\}))$ achieves this effect

Equivalence of NFA and DFA

- We have now completed the construction of the DFA M that simulates the NFA N.
- The construction of M obviously works correctly.
- At every step in the computation of M on an input, it clearly enters a state that corresponds to the subset of states that N could be in at that point.
- Thus our proof is complete.

Example



An NFA accepting all strings that end in 01

Example-continued

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

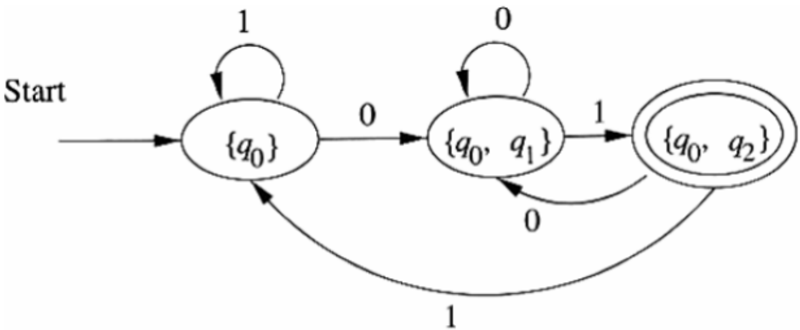
Figure 2.12: The complete subset construction from Fig. 2.9

Example-continued

	0	1
A	A	A
$\rightarrow B$	E	B
C	A	D
$*D$	A	A
E	E	F
$*F$	E	B
$*G$	A	D
$*H$	E	F

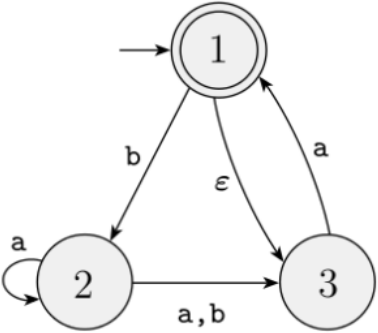
Renaming the states

Example-continued

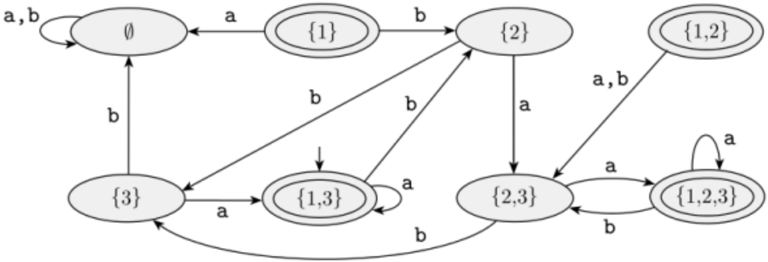


The DFA constructed from the NFA

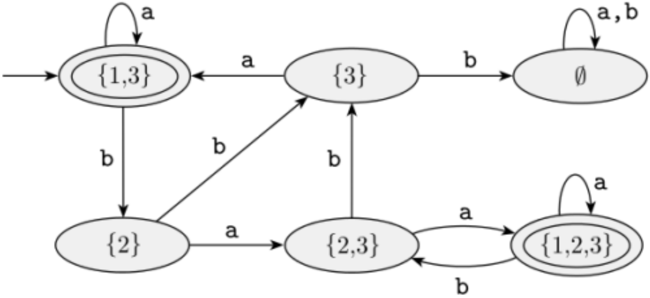
Example-2



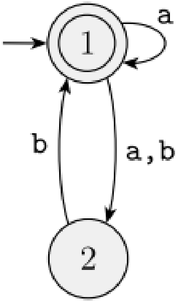
Example-2 continued



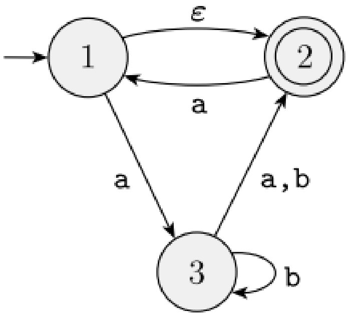
Example-2 continued



Exercise-1 Convert the following NFA to equivalent DFA



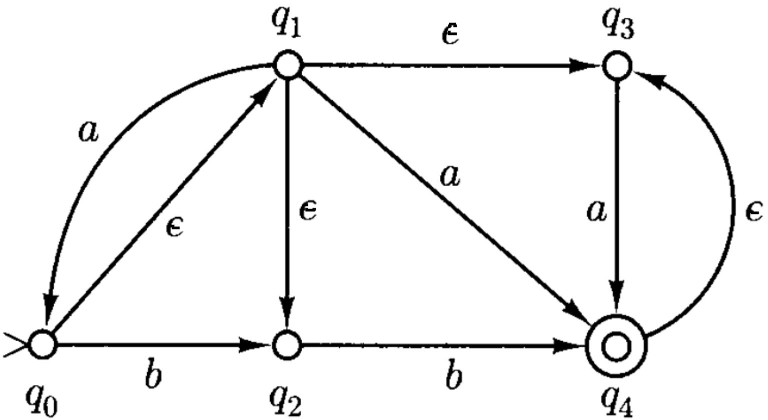
Exercise-2 Convert the following NFA to equivalent DFA



Example

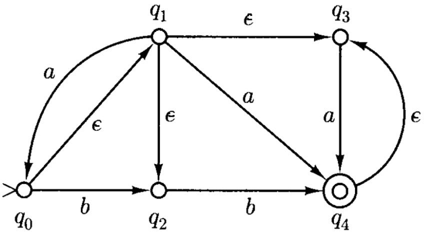
Lewis and Papadimitriou, Example 2.2.3, p-70

We find the DFA equivalent to the nondeterministic automaton.



$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', q_0{}', F')$$

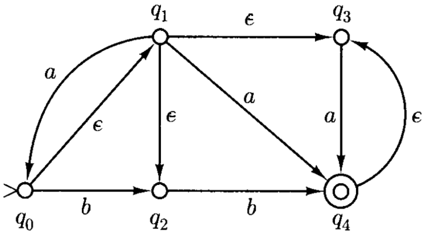
- Q' is the power set of Q.



- Since N has 5 states, D will have $2^5 = 32$ states.
- However, only a few of these states will be relevant to the operation of D.

$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', q_0', F')$$

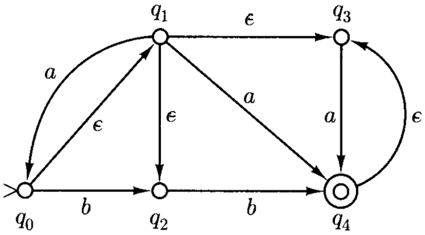
- Q' is the power set of Q.



- Namely, those states that can be reached from state q_0' by reading some input string.
- Obviously, any state in D that is not reachable from q_0' is irrelevant to the operation of D.

$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', q_0', F')$

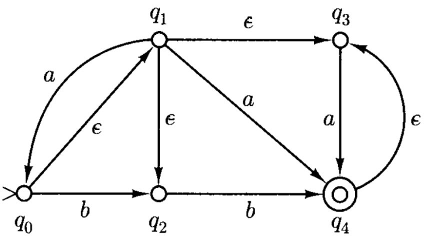
- Q' is the power set of Q.



- We shall build this by *lazy evaluation* on the subsets.

$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', q_0', F')$

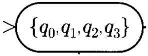
- $q_0{}' = E(q_0)$.



- $q_0{}' = E(q_0) = \{q_0, q_1, q_2, q_3\}$.

$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', q_0', F')$$

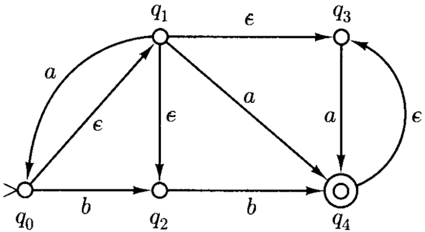
- $q_0' = E(q_0)$.



- $q_0' = E(q_0) = \{q_0, q_1, q_2, q_3\}$.

Example — *continued*

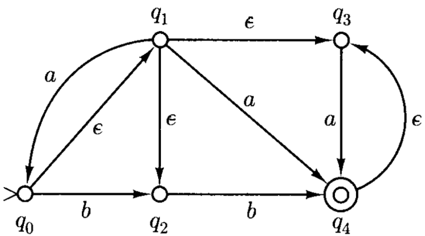
$N = (Q, \Sigma, \delta, q_0, F)$ $\qquad$ $D = (Q', \Sigma, \delta', q_0', F')$



- $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) = \emptyset \cup \{q_0, q_4\} \cup \emptyset \cup \{q_4\} = \{q_0, q_4\}$

Example — *continued*

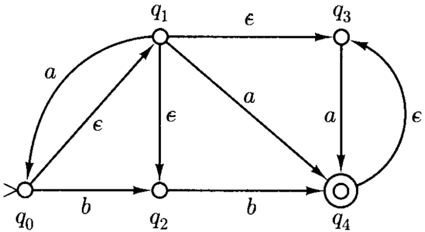
$N = (Q, \Sigma, \delta, q_0, F)$ $\quad$ $D = (Q', \Sigma, \delta', {q_0}', F')$



- $E(q_0) = \{q_0, q_1, q_2, q_3\}$, and $E(q_4) = \{q_3, q_4\}$.
- $\delta'({q_0}', a) = \{q_0, q_1, q_2, q_3\} \cup \{q_3, q_4\} = \{q_0, q_1, q_2, q_3, q_4\}$.

Example — *continued*

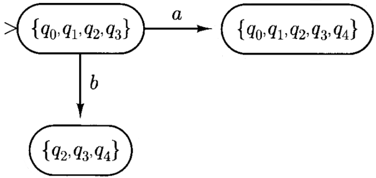
$N = (Q, \Sigma, \delta, q_0, F)$ $\quad\quad$ $D = (Q', \Sigma, \delta', q_0', F')$



- Similarly, $\delta'(q_0', b) = \{q_2, q_3, q_4\}$.

Example — *continued*

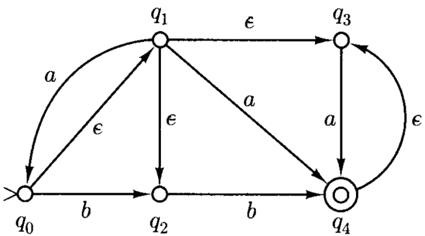
$N = (Q, \Sigma, \delta, q_0, F)$ $\qquad$ $D = (Q', \Sigma, \delta', q_0', F')$



- Similarly, $\delta'(q_0', b) = \{q_2, q_3, q_4\}$.

Example — *continued*

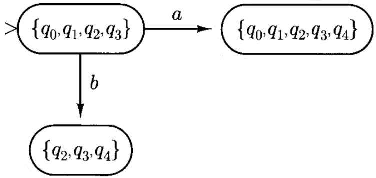
$N = (Q, \Sigma, \delta, q_0, F)$ $\quad$ $D = (Q', \Sigma, \delta', q_0', F')$



- We repeat the calculation for the newly introduced states.
- $\delta'(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$, and
- $\delta'(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}$.

Example — *continued*

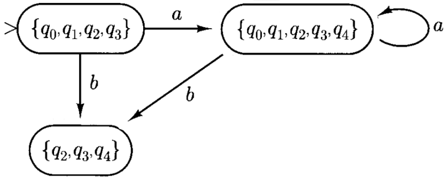
$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', q_0{}', F')$$



- We repeat the calculation for the newly introduced states.
- $\delta'(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$, and
- $\delta'(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}$.

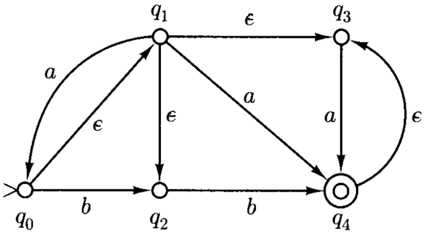
Example — *continued*

$N = (Q, \Sigma, \delta, q_0, F)$ $\qquad$ $D = (Q', \Sigma, \delta', q_0{}', F')$



- We repeat the calculation for the newly introduced states.
- $\delta'(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$, and
- $\delta'(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}$.

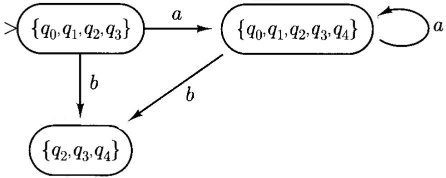
$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', {q_0}', F')$



- Also we get.
- $\delta'(\{q_2, q_3, q_4\}, a) = \{q_3, q_4\}$, and
- $\delta'(\{q_2, q_3, q_4\}, b) = \{q_3, q_4\}$.

Example — *continued*

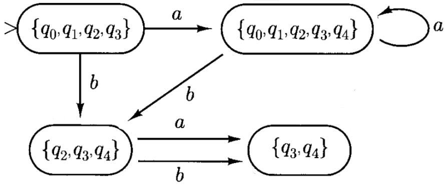
$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', q_0{}', F')$



- Also we get.
- $\delta'(\{q_2, q_3, q_4\}, a) = \{q_3, q_4\}$, and
- $\delta'(\{q_2, q_3, q_4\}, b) = \{q_3, q_4\}$.

Example — *continued*

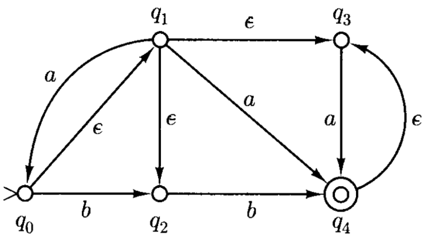
$N = (Q, \Sigma, \delta, q_0, F)$ $\qquad$ $D = (Q', \Sigma, \delta', q_0{}', F')$



- Also we get.
- $\delta'(\{q_2, q_3, q_4\}, a) = \{q_3, q_4\}$, and
- $\delta'(\{q_2, q_3, q_4\}, b) = \{q_3, q_4\}$.

Example — *continued*

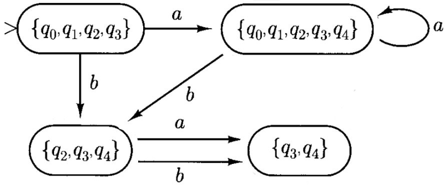
$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', q_0{}', F')$



- Next we get.
- $\delta'(\{q_3, q_4\}, a) = \{q_3, q_4\}$, and
- $\delta'(\{q_3, q_4\}, b) = \emptyset$.

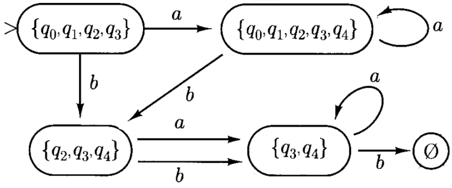
Example — *continued*

$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', q_0', F')$



- Next we get.
- $\delta'(\{q_3, q_4\}, a) = \{q_3, q_4\}$, and
- $\delta'(\{q_3, q_4\}, b) = \emptyset$.

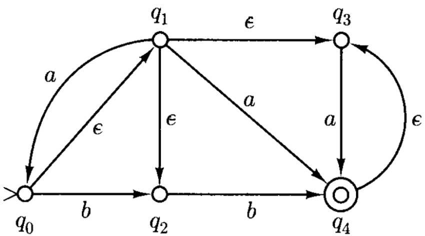
$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', q_0{}', F')$



- Next we get.
- $\delta'(\{q_3, q_4\}, a) = \{q_3, q_4\}$, and
- $\delta'(\{q_3, q_4\}, b) = \emptyset$.

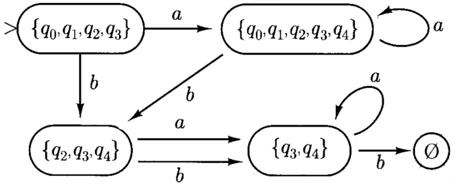
Example — *continued*

$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', q_0', F')$



- Finally, we get.
- $\delta'(\emptyset, a) = \delta'(\emptyset, b) = \emptyset$.

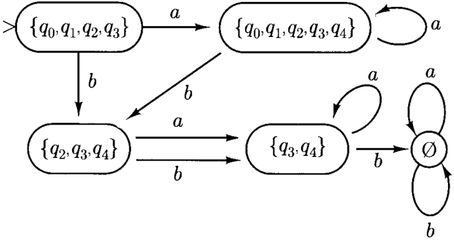
$N = (Q, \Sigma, \delta, q_0, F)$ $D = (Q', \Sigma, \delta', {q_0}', F')$



- Finally, we get.
- $\delta'(\emptyset, a) = \delta'(\emptyset, b) = \emptyset$.

Example — *continued*

$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', {q_0}', F')$$

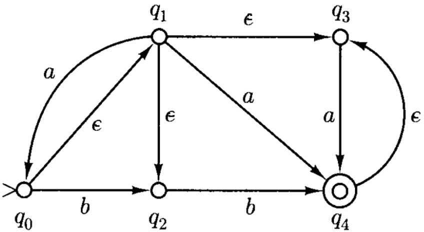


- Finally, we get.
- $\delta'(\emptyset, a) = \delta'(\emptyset, b) = \emptyset$.

Example — *continued*

$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', q_0', F')$$

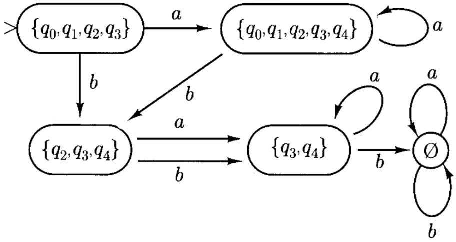
- F' is those sets of states that contain at least one accepting state of N.



- q_4 is the sole member of F.
- The set of final states, contains each set of states of which q_4 is a member.

$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', q_0{}', F')$$

- F' is those sets of states that contain at least one accepting state of N.

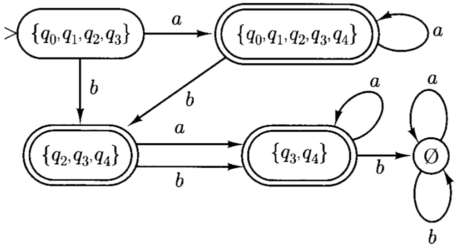


- q_4 is the sole member of F.
- The set of final states, contains each set of states of which q_4 is a member.

Example — *continued*

$$N = (Q, \Sigma, \delta, q_0, F) \qquad D = (Q', \Sigma, \delta', q_0', F')$$

- F' is those sets of states that contain at least one accepting state of N.



- The three states $\{q_0, q_1, q_2, q_3, q_4\}$, $\{q_2, q_3, q_4\}$, and $\{q_3, q_4\}$ are final.

Equivalence of NFAs AND DFAs

Sipser, 1.2, p-56

COROLLARY 1.40

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

Closure under the Regular Operations

Sipser, 1.2, p-59

THEOREM 1.45

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

Closure under the Regular Operations

Sipser, 1.2, p-59

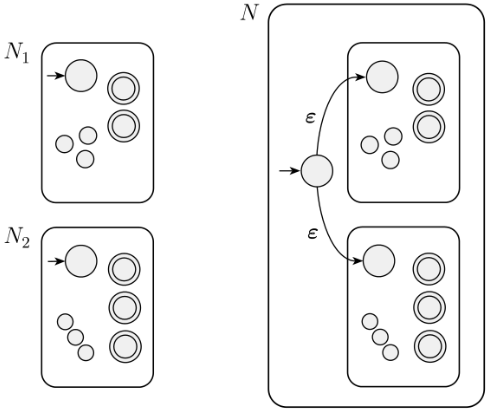
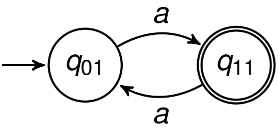


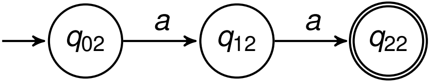
FIGURE 1.46
Construction of an NFA N to recognize $A_1 \cup A_2$

- $L_1 = \{\text{contains an odd number of } a\text{'s}\}$
  $L_2 = \{aa\}$

N_1



N_2



PROOF

- Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.
- And $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
- Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.

- The states of N are all the states of N_1 and N_2, with the addition of a new start state q_0.

2. The state q_0 is the start state of N.

Equivalence of NFAs AND DFAs — *continued*

Sipser, 1.2, p-56

3. The set of accept states $F = F_1 \cup F_2$.

- The accept states of N are all the accept states of N_1 and N_2.
- That way, N accepts if either N_1 accepts or N_2 accepts.

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

Closure under the Regular Operations

Sipser, 1.2, p-60

THEOREM 1.47 ....

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

Closure under the Regular Operations

Sipser, 1.2, p-60

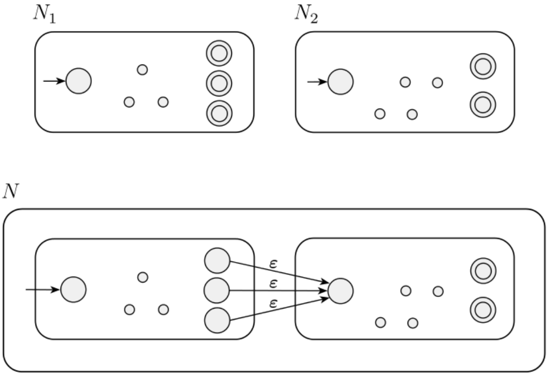
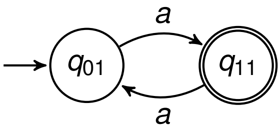


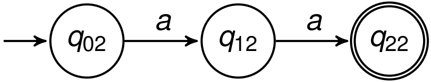
FIGURE 1.48
Construction of N to recognize $A_1 \circ A_2$

- $L_1 = \{$contains an odd number of a's$\}$
  $L_2 = \{aa\}$

N_1



N_2



PROOF

- Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.
- And $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
- Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \circ A_2$.

Equivalence of NFAs AND DFAs — *continued*

Sipser, 1.2, p-61

1. $Q = Q_1 \cup Q_2$.

- The states of N are all the states of N_1 and N_2.

2. The state q_1 is the start state of N.

Equivalence of NFAs AND DFAs — *continued*

Sipser, 1.2, p-61

3. The set of accept states $F = F_2$.

- The accept states F are the same as the accept states of N_2.

Equivalence of NFAs AND DFAs — *continued*

Sipser, 1.2, p-61

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a) & q \in Q_2 \end{cases}$$

Closure under the Regular Operations

Sipser, 1.2, p-62

THEOREM 1.49

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

Closure under the Regular Operations

Sipser, 1.2, p-62

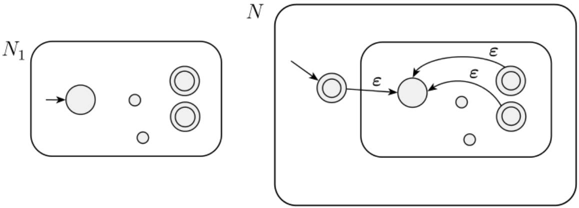
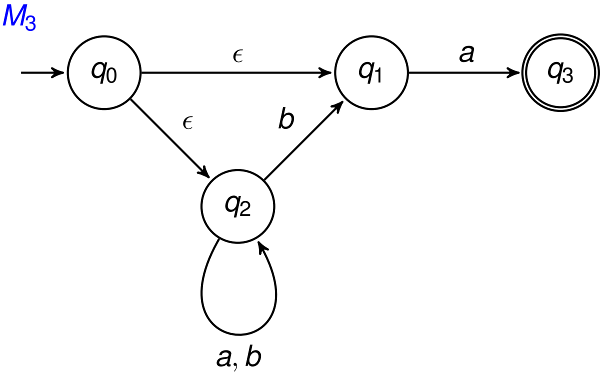


FIGURE 1.50
Construction of N to recognize A^*

- $\sum = \{a, b\}$, $L_3 = \{$ends in exactly one a at the end$\}$

- $\sum = \{a, b\}$, $L_3 = \{\text{ends in exactly one } a \text{ at the end}\}$



PROOF

- Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.
- Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

Equivalence of NFAs AND DFAs — *continued*

Sipser, 1.2, p-62

1. $Q = \{q_0\} \cup Q_1$.

- The states of N are the states of N_1 plus a new start state.

2. The state q_0 is the new start state.

Equivalence of NFAs AND DFAs — *continued*

Sipser, 1.2, p-62

3. $F = \{q_0\} \cup F_1$.

- The accept states are the old accept states plus the new start state.

Equivalence of NFAs AND DFAs — *continued*

Sipser, 1.2, p-62

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & q \in q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$