FINITE AUTOMATA

Chap. 2

Summary

- The *simple* automaton, a finite state accepter:
  - a *finite set of internal states* and *no other memory*.
  - It process strings and either *accepts* or *reject* them.
  - A simple pattern recognition mechanism.
- Deterministic Finite Automaton/Accepter (DFA)
  - Deterministic: the automaton has only one transition to a next state per a symbol at one time.
  - Regular language
- Nondeterministic Finite Automaton/Accepter (NFA)
  - Nondeterministic: several transitions for a choice of next state.
  - NFA explores all choices and makes no decision until all options have been analyzed.
  - NFA simplifies the solution of many problems.
- Equivalence of DFA & NFA

Learning Objectives

- Describe the components of a *Deterministic Finite Automata* (DFA).
- State whether an input string is *accepted* by a DFA.
- Describe the *language* accepted by a DFA.
- Construct a DFA to accept a specific language.
- Show that a particular language is regular.
- Describe the *differences* between *DFA* and *NFA*.
- State whether an input string is accepted by a NFA.
- Construct a NFA to accept a specific language.
- Transform an arbitrary NFA to an equivalent DFA.

Deterministic Finite Automata

- **Formal** Definition 2.1: A *Deterministic Finite Automata, DFA, (or accepter, recognizer)* is defined by the quintuple

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

where

Q : a *finite* set of *internal states*
Σ : a set of symbols, called the *input alphabet*
$\delta: Q \times \Sigma \to Q$ -- a *transition function*
$q_0 \in Q$: the *initial state*
F ($\subseteq$ Q) : a set of the *final states*.

- Example 2.1: Consider the DFA

$Q = \{ q_0, q_1, q_2 \}$, $\Sigma = \{ 0, 1 \}$, $F = \{ q_1 \}$

where the transition function is given by

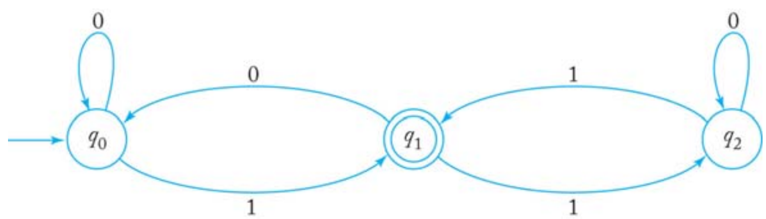
$\{ \delta(q_0, 0) = q_0, \ \delta(q_0, 1) = q_1, \ \delta(q_1, 0) = q_0$
$\delta(q_1, 1) = q_2, \ \delta(q_2, 0) = q_2, \ \delta(q_2, 1) = q_1 \}$.

Transition Diagram/Graph

- A transition function of DFA can be visualized with a *Transition Diagram.*
- Example 2.1: M = (Q, Σ, δ, q_0, F) where

  Q = { q_0, q_1, q_2 }, Σ = { 0, 1 }, F = { q_1 }

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



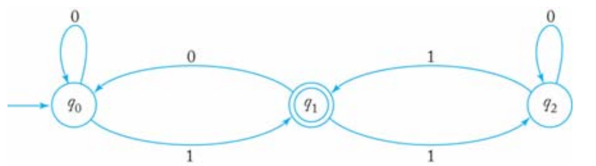
Processing Input with a DFA

- A DFA starts by processing the leftmost input symbol with its control in state q_0. The transition function determines the next state, based on current state and input symbol.
- The DFA continues processing input symbols until the end of the input string is reached.
- The input string is *accepted* if the automaton is in a final state after the last symbol is processed. Otherwise, the string is *rejected*.
- For example, the DFA in Ex. 2.1 accepts the string 111 but rejects the string 110.

The Language Accepted by a DFA

- For a given DFA, the *extended transition function* δ^* accepts a DFA state and an *input string* as input. The value of the function is the state of the automaton after the string is processed: $\delta^*: Q \times \Sigma^* \rightarrow Q$, s.t.
  - $\delta^*(q, \lambda) = q$,
  - $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$, $w \in \Sigma^*$, $a \in \Sigma$
- Sample values of δ^* for the DFA in Ex. 2.1,

  $\delta^*(q_0, 1001) = q_1 (\in F)$, $\delta^*(q_1, 000) = q_0 (\notin F)$



The Language Accepted by a DFA (cont.)

- Def 2.2: *The language accepted by a DFA* M = (Q, Σ, δ, q_0, F) is the *set of all strings* on Σ accepted by M, i.e. the set of all strings w whose transition results in a final state.

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

- Note: The language *rejected* by a DFA M is

  $\overline{L(M)} = \{w \in \Sigma^* \mid \delta^*(q_0, w) \notin F\}$

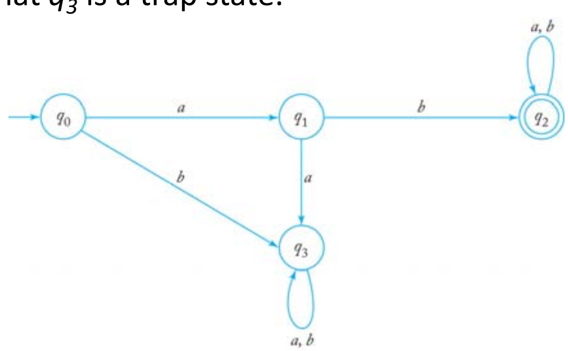
Example: DFA (Acceptor, Recognizer)

- Example 2.2: a DFA to accept the set of all strings on $\Sigma=\{a, b\}$, consisting of an arbitrary number of *a*'s, followed by a single *b*, i.e. $L = \{a^n b \mid n \geq 0\}$.
- Note that q_2 is a trap state.

	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

Example: DFA (Acceptor, Recognizer)

- Example 2.3: a DFA to accept the set of all strings on $\{a, b\}$ that start with the prefix *ab*, i.e. $\{ abw \mid w \in \{a, b\}^* \}$
- Note that q_3 is a trap state.

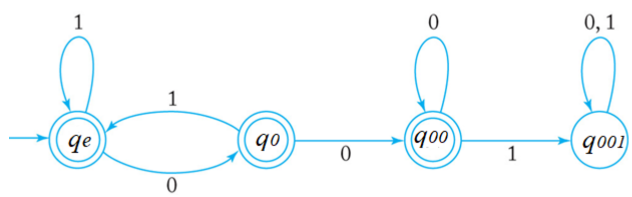


Example: DFA (Acceptor, Recognizer)

- Example 2.4: a DFA M' to accept all strings on $\Sigma=\{0, 1\}$, *except* those containing the substring 001.

  i.e. L(M) = { $v001w$ | $\delta^*(q_0, v001w) \in F$ $\forall v, w \in \{0,1\}^*$ }

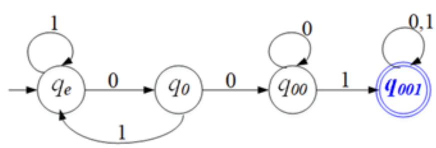
  → L(M') = $\overline{\text{L(M)}}$= { u | $\delta^*(q_0, v001w) \notin F$, $\forall u, v, w \in \{0,1\}^*$ }
- Note that any state except *a state 'q001'* is a final state.



Example: cont.

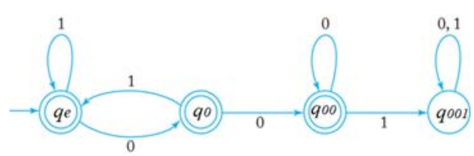
- Example 2.4A: a DFA M to accept all strings on $\Sigma=\{0, 1\}$, containing the substring 001.

  i.e. L(M) = { $v001w$ | $\delta^*(q_0, v001w) \in F$ $\forall v, w \in \{0,1\}^*$ }



Example 2.4B: a DFA M' to accept all strings on $\Sigma=\{0, 1\}$, *except* those containing the substring 001.

→ L(M') = $\overline{\text{L(M)}}$= { u | $\delta^*(q_0, v001w) \notin F$, $\forall u, v, w \in \{0,1\}^*$ }



Regular Languages

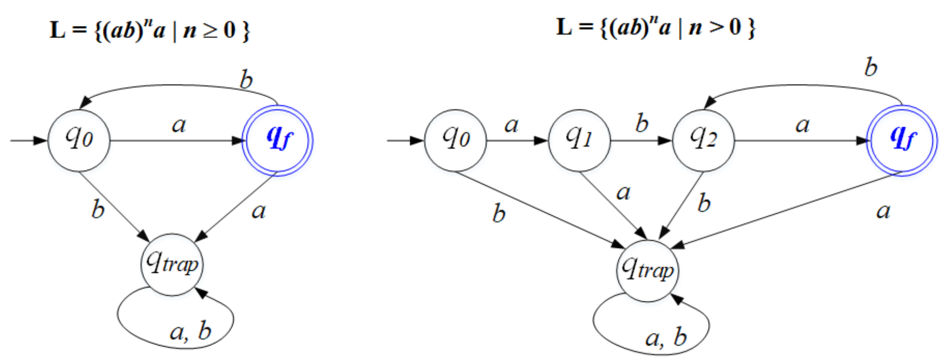
- Finite automata accept a family of languages collectively known as *regular languages*.
- Def. 2.3: A language L is *regular* if and only if

  there exists a DFA, M, that accepts L, i.e. $L = L(M)$.
- To show that a language is regular,

  one must construct a DFA to accept it.
- Example: Show that $L = \{(ab)^n a \mid n > 0\}$ is regular.

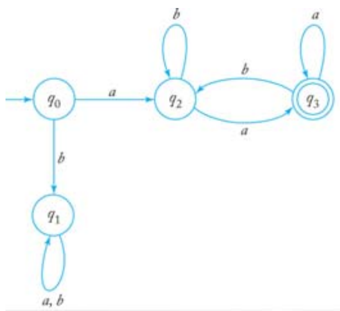
  → Construct a DFA, M, that accepts L, i.e. $L = L(M)$.

  $L = \{aba,\ ababa,\ abababa,\ \ldots\ldots\}$
- Regular languages have wide applicability in problems that involve scanning input strings in search of specific patterns.

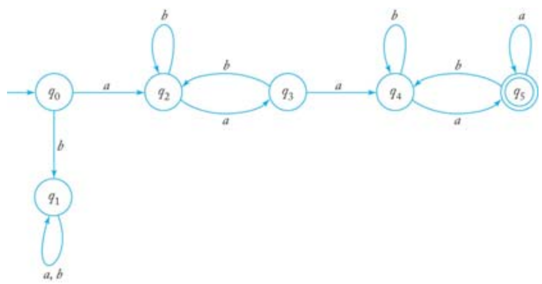


Example: Regular Languages

- Example 2.5:
  Show that the language
  $L=\{awa \mid w \in \{a,b\}^*\}$
  is *regular*.



- Example 2.6:
  Show the language $L^2 = LL$ is regular,
  $L^2=\{aw_1aaw_2a \mid w_1,w_2 \in \{a,b\}^*\}$.



where q_1 is a trap state.

Nondeterministic Finite Automaton (NFA)

- An automaton is *nondeterministic* if it has a choice of actions for given inputs.
- Def 2.4: A *Nondeterministic Finite Automata, NFA, (or accepter, recognizer)* is defined by the quintuple

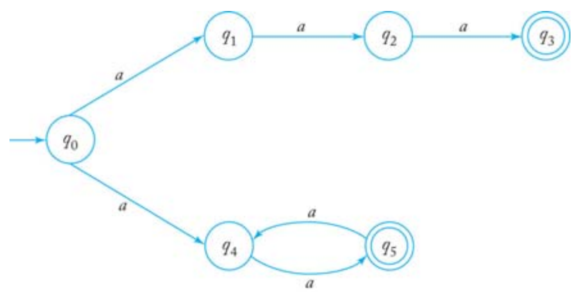
  $$M = (Q, \Sigma, \delta, q_0, F)$$

  where Q, Σ, q_0, F are defined as for DFA, but
  $\delta: Q \times (\Sigma \cup \lambda) \rightarrow 2^Q$ -- a *transition function*
- Basic Differences between DFA and NFA:
  1. In an NFA, a transition of (state, symbol) may lead to *several states* simultaneously.
  2. If a transition is labeled with the empty string (λ) as its input symbol, the NFA *may change states* without consuming input (i.e. with λ): λ-transition
  3. An NFA may have *undefined transitions.*

Example: Nondeterministic FA

- Example 2.7: a Nondeterministic FA in which there are two transitions labeled '*a*' out of state q_0:

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

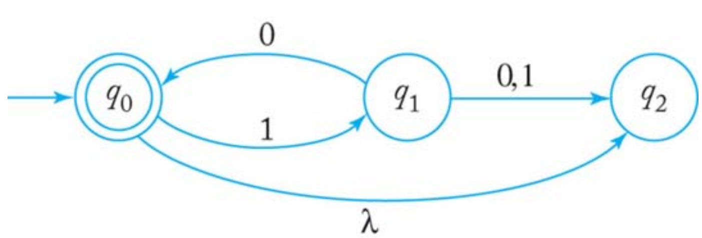


Example: Nondeterministic FA

- Example 2.8: A NFA which contains both a λ-transition as well as undefined transitions:

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

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

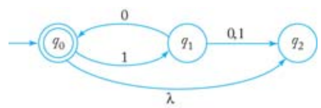


The Language Accepted by a NFA

- For a given NFA, the value of the *extended transition* function $\delta^*(q_i, w)$ is the set of all possible states for the control unit after processing w, having started in q_i.
- Sample values of δ^* for the NFA in Ex. 2.8:

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

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



- A string w is accepted if $\delta^*(q_0, w)$ contains a final state.

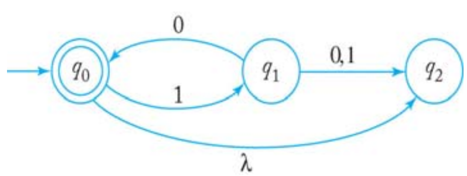
  i.e. $\delta^*(q_0, w) \cap F \neq \varnothing$.
- In Ex 2.8, 10 would be accepted but 101 would be rejected.
- *The language accepted by a NFA* M is the set of all accepted strings: $L(M) = \{ w \mid \delta^*(q_0, w) \cap F \neq \varnothing \}$.
- The NFA in Ex. 2.8 accepts $L = \{ (10)^n \mid n \geq 0 \}$.

Equivalence of DFA and NFA

- Def. 2.7: Two FAs, M_1 and M_2, are said to be *equivalent* if

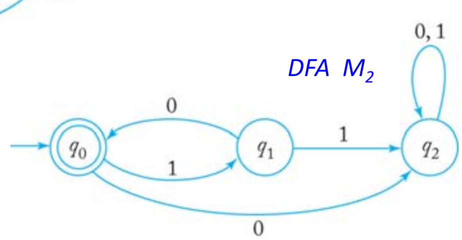
  $L(M_1) = L(M_2)$

  i.e. if they both accept the same language.
- Ex. 2.11: $L = \{ (10)^n \mid n \geq 0 \}$



NFA M_1

DFA M_2



$M_1 \equiv M_2$

Equivalence of DFA and NFA

- Does Nondeterminism make it possible to accept languages that DFA cannot recognize?
- Theorem 2.2:

  Let L be the language accepted by a NFA $M_N = (Q, \Sigma, \delta, q_0, F)$.

  Then, there exists a DFA $M_D = (Q', \Sigma, \delta', q_0', F')$

  where $Q' \subseteq \mathcal{P}(Q)$, $q_0' = \{q_0\}$,

  $F' \subseteq \mathcal{P}(q_F)$, *for any* $q_F \in F$,

  i.e. $F' = \{q' \in Q' \mid q'$ contains an accept state of N$\}$,

  such that $L = L(M_D) = L(M_N)$

  i.e. *For any NFA, there is an equivalent DFA that accepts the same language.*
- Therefore, every language accepted by a NFA is regular.
- Proof) A *constructive proof* for Thm 2.2:

  The algorithm of building a DFA equivalent to a particular NFA.

Procedure: NFA with no λ-transition -to-DFA Conversion

1. Create DFA with an initial start state $\{q_0\}$.

   Beginning with the start state q_0 in NFA, define input transitions for the DFA as follows:
2. Repeat
   - If the NFA input transition to a single state q_k,
     replicate it for the DFA $\{q_k\}$.
   - If the NFA input transition leads to more than one state,
     create a new state in the DFA labeled $\{ q_i, \ldots, q_j \}$,
     where $q_i, \ldots, q_j$ are the states to which the NFA transition can lead.
   - If the NFA input transition is not defined,
     the corresponding DFA transition should lead to a trap state.

   Until no new states are created for all newly created DFA states.
3. *Any* DFA state containing any $q_F \in F$ is labeled as a final state.
4. (If the NFA accepts λ, label the start state of DFA as a *final state)*.

λ-closure (or ε-closure) – not in the textbook

Prior to construct an equivalent DFA from a NFA,
let's define a λ-closure (or ε-closure) of a state for Q'.

- For any $q' \in Q' \subseteq \mathcal{P}(Q)$, the λ-closure of q' is
  $\Lambda(q')$ (= $E(q')$)
  $= q' \cup \{ q \in Q \mid q$ is reachable from q' through 0 or more λ-transitions$\}$
  $= q' \cup \{q \in Q \mid \forall r_1 \in q', \exists r_2, \ldots, r_k \in Q, r_{i+1} \in \delta(r_i, \lambda), r_k = q\}$
  i.e. the collection of states that are reachable from q' through *0* or *more* λ-transitions, including the members of q' themselves.
- DFA M_D is such that
  1. $Q' \subseteq \mathcal{P}(Q)$, (i.e. $q' \subseteq Q \Leftrightarrow \forall q' \in Q'$)
  2. For $q' \in Q'$ and $a \in \Sigma$, $\delta'(q', a) = \bigcup_{r \in q', q' \subseteq Q} \Lambda(\delta(r, a))$
  3. $q_0' = \Lambda\{q_0\}$
  4. $F' = \{q' \in Q' \mid q' \cap F \neq \emptyset$, i.e. q' contains a final state of an NFA $M_N\}$
     $\cup \{q_0' \mid$ if N accepts $\lambda\}$

Procedure: NFA with λ-transition -to-DFA Conversion

1. Create DFA with an initial start state, $q_0' = \Lambda\{q_0\}$
   Beginning with the start state of q_0 in NFA,
   define input transitions for the DFA as follows:
2. Repeat
   - If the NFA input transition to a single state q_k,
     replicate it for the DFA $q_k' = \Lambda(\{q_k\})$.
   - If the NFA input transition leads to more than one state,
     create a new state in the DFA labeled $\{ q_i, \ldots, q_j \}$,
     where $q_i, \ldots, q_j$ are the states to which the NFA transition
     with its λ-closure can lead : i.e. $\delta'(q', a) = \bigcup_{r \in q', q' \subseteq Q} \Lambda(\delta(r, a))$
   - If the NFA input transition is not defined,
     the corresponding DFA transition should lead to a trap state.

   Until no new states are created for all newly created DFA states.
3. *Any* DFA state containing any $q_F \in F$ is labeled as a final state.
4. If the NFA accepts λ, label the start state of DFA as a *final state*.

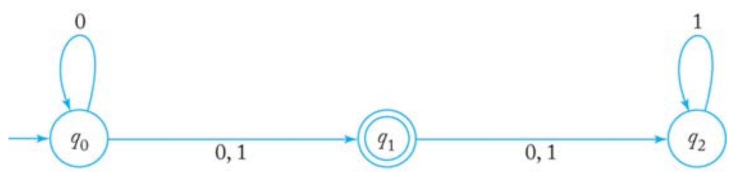
Example 2.13: NFA-to-DFA Conversion
-- NFA without λ-transition

- When applying the conversion procedure to the NFA below, we note the following NFA transitions

$\delta_N(q_0, 0) = \{ q_0, q_1 \}$ $\quad$ $\delta_N(q_0, 1) = \{ q_1 \}$

$\delta_N(q_1, 0) = \{ q_2 \}$ $\quad$ $\delta_N(q_1, 1) = \{ q_2 \}$

$\delta_N(q_2, 0) = \varnothing$ $\quad$ $\delta_N(q_2, 1) = \{ q_2 \}$



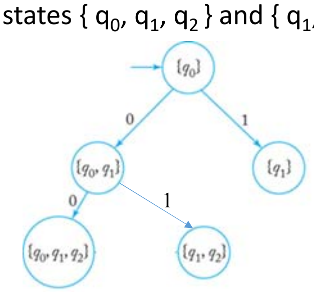
Example: NFA-to-DFA Conversion (cont.)

- Add a new start state $\{q_0\}$ in M_D.
- For a new $\{q_0\}$,
  - note that $\delta_N(q_0,0) = \{q_0, q_1\}$ and $\delta_N(q_0,1) = \{q_1\}$.
  - So, add transitions from $\{q_0\}$ to states $\{ q_0, q_1 \}$ and $\{ q_1 \}$:

    $\delta_D(\{q_0\}, 0) = \{ q_0, q_1 \}$ and $\delta_D(\{q_0\}, 1) = \{q_1\}$.
- For a new $\{q_0, q_1\}$,
  - note that $\delta_N(q_0,0) \cup \delta_N(q_1,0) = \{q_0, q_1, q_2\}$
    and $\delta_N(q_0,1) \cup \delta_N(q_1,1) = \{q_1, q_2\}$.
  - So, add transitions from $\{q_0, q_1\}$ to states $\{ q_0, q_1, q_2 \}$ and $\{ q_1, q_2 \}$:

    $\delta_D(\{q_0, q_1\}, 0) = \{ q_0, q_1, q_2 \}$

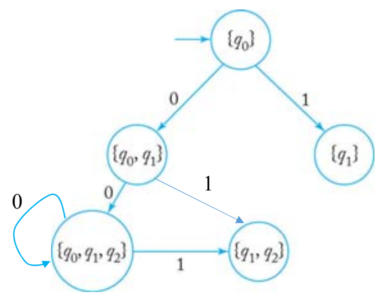
    and $\delta_D(\{q_0, q_1\}, 1) = \{q_1, q_2 \}$



Example: NFA-to-DFA Conversion (cont.)

- For a new $\{q_0, q_1, q_2\}$,
  - note that $\delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0) = \{q_0, q_1, q_2\}$
    and $\delta_N(q_0, 1) \cup \delta_N(q_1, 1) \cup \delta_N(q_2, 1) = \{ q_1, q_2 \}$
  - So, add transitions from $\{q_0,q_1,q_2\}$ to states $\{q_0,q_1,q_2\}$ and $\{q_1,q_2 \}$:

    $\delta_D(\{q_0,q_1,q_2 \}, 0) = \{q_0,q_1,q_2\}$ and $\delta_D(\{q_0,q_1,q_2 \}, 1) = \{q_1, q_2 \}$.

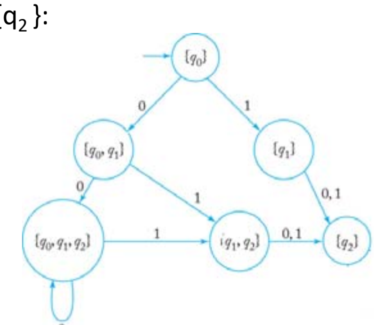


Example: NFA-to-DFA Conversion (cont.)

- For a new $\{q_1\}$,
  - note that $\delta_N(q_1, 0) = \{q_2\}$ and $\delta_N(q_1, 1) = \{q_2\}$.
  - So, add 0-1 transitions from $\{q_1\}$ to state $\{q_2 \}$:

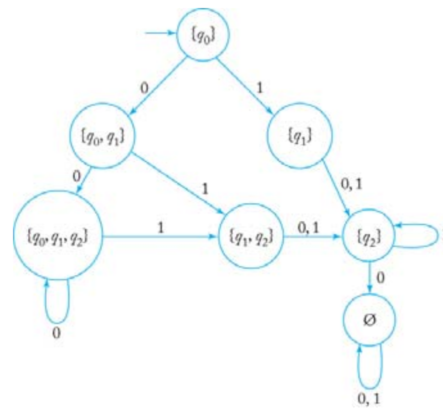
    $\delta_D(\{q_1 \}, 0) = \delta_D(\{q_1\}, 1) = \{q_2 \}$.
- For a new $\{q_1 , q_2\}$,
  - note that $\delta_N(q_1, 0) \cup \delta_N(q_2, 0) = \{q_2\}$ and $\delta_N(q_1, 1) \cup \delta_N(q_2, 1) = \{q_2\}$.
  - So, add transitions from $\{q_1,q_2\}$ to state $\{q_2 \}$:

    $\delta_D(\{q_1,q_2 \}, 0) = \delta_D(\{q_1,q_2 \}, 1) = \{q_2 \}$.



Example: NFA-to-DFA Conversion (cont.)

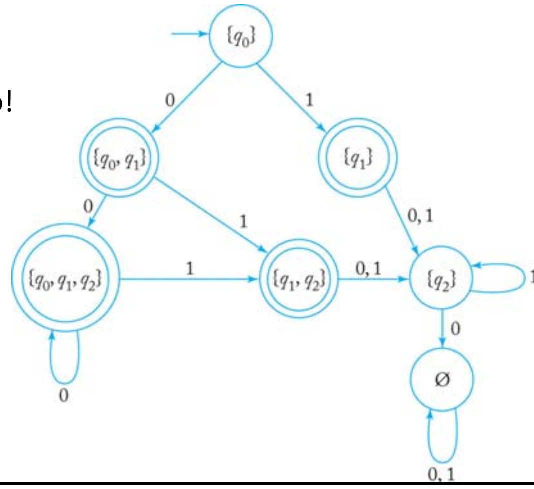
- For a new $\{q_2\}$,
  - add a transition $\delta_D(\{q_2\}, 1) = \{q_2\}$ since $\delta_N(q_2, 1) = \{q_2\}$.
  - However, since $\delta_N(q_2, 0)$ is undefined, we add a trap state (labeled $\varnothing$) for a transition with 0 from $\{q_2\}$: i.e. $\delta_D(\{q_2\}, 0) = \varnothing$.
- For a trap state $\varnothing$,
  - add the dummy transitions to itself:
    $\delta_D(\varnothing, 0) = \delta_D(\varnothing, 1) = \varnothing$
- Since there are no DFA states with undefined transitions, the process stops.
- All states containing $q_1(=q_F)$ in their label are designated as final states.



Example: NFA-to-DFA Conversion (cont.)

- Mark the new final states:

  All states containing $q_1(=q_F$ of NFA) in their label are designated as final states.



- $\lambda \in L(M_N)$? No!
  So, $\{q_0\} \notin F_D$

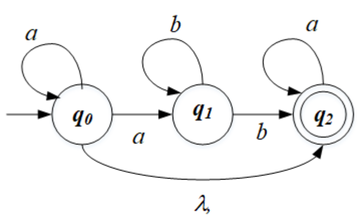
Example: NFA with λ-transition -to-DFA Conversion

- Convert the NFA defined by the transitions below with the initial state q_0 and the final state q_2 into an *equivalent DFA*.

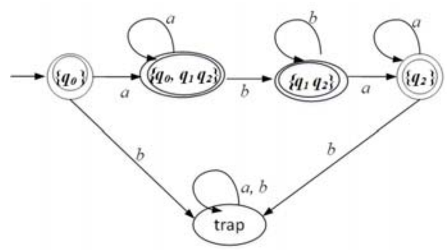
  Draw the transition graph of the DFA.

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

NFA



DFA ?



Reduction of the Number of States in DFA

- Construct a FA with the *minimum # of states*.
- The computation requires space/time proportional to the # of states → Reduce the # of states for the storage/time efficiency.
- Definition 2.8:

  Two states p and q of a DFA are *indistinguishable* if

  $\delta^*(p, w) \in F$ implies $\delta^*(q, w) \in F$,

  and $\delta^*(p, w) \notin F$ implies $\delta^*(q, w) \notin F$, $\forall w \in \Sigma^*$.

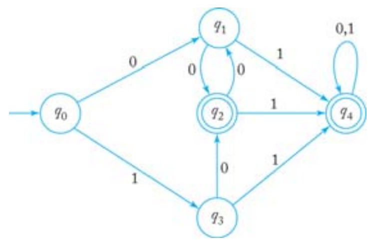
  If, on the other hand, there exists some string $w \in \Sigma^*$ s.t.

  $\delta^*(p, w) \in F$ and $\delta^*(q, w) \notin F$, or vice versa,

  then, the states p and q are said to be *distinguishable* by a string w.

Reduction of the # of States (cont.)

- Example 2.15: Mark procedure



- Partition the states into a class of final states and that of non-finals:
  $\{q_0, q_1, q_3\}$ and $\{q_2, q_4\}$
- For *all* pair of states in the same class, mark the distinguishable:
  - For (q_0, q_1): $\delta(q_0, 0)= q_1$ and $\delta(q_1, 0) = q_2$: So, q_0, q_1 are distinguishable.
  - For (q_1, q_3): $\delta(q_1, 0)= \delta(q_3, 0) = q_2$: So, q_1, q_3 are indistinguishable.
  - For (q_2, q_4): $\delta(q_2, 0)= q_1$ and $\delta(q_4, 0) = q_4$: So, q_2, q_4 are distinguishable.
  - So, the equivalence classes are $\{q_0\}$, $\{q_1, q_3\}$, $\{q_2\}$ and $\{q_4\}$.

Reduction of the # of States (cont.)

- Procedure: Reduce.

  Given a DFA $M=(Q,\Sigma,\delta,q_0,F)$,

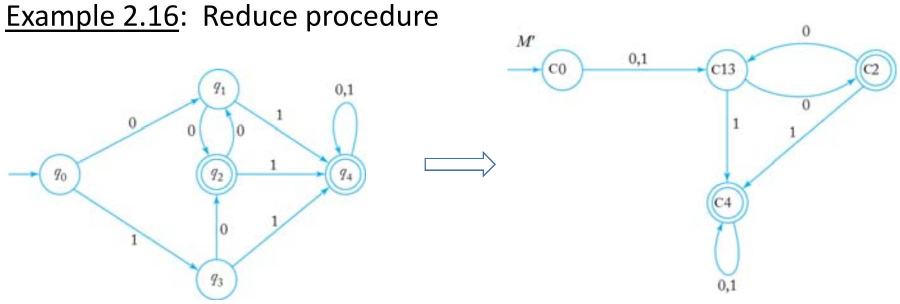
  we construct a *reduced DFA* $M' =(Q', \Sigma, \delta', q_0', F')$

  where $Q' \in \wp(Q)$, $\delta': Q' \times \Sigma \rightarrow Q'$, $q_0'=\{q_0\}$, $F' \subseteq \wp(q_F)$

  1. Generate the equivalence classes by Mark procedure.
  2. For each class C_i, create a state labeled C_i for M'.
  3. For each transition rule of M, $\delta(q_r, a)= q_p$,
     find the classes C_r and C_p to which q_r and q_p belong, resp.,
     then, add to δ' a rule: $\delta'(C_r, a)= C_p$.
  4. The initial state q_0' is the labeled equiv. class which includes q_0.
  5. F' is the set of all the labeled equiv. states which contains $q_F \in F$.

Reduction of the # of States (cont.)

- Example 2.16: Reduce procedure

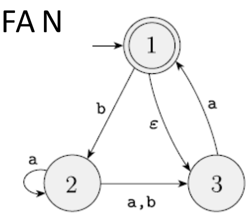


From the equivalence classes C_0={q_0}, C_{13}={q_1,q_3}, C_2={q_2} and C_4= {q_4} in Ex. 2.15.

- For $\delta(q_0, 0)= q_1$ *and* $\delta(q_0, 1)= q_3$, add a transition $\delta'(C_0, 0)= \delta'(C_0, 1)= C_{13}$ to *M'*.
- For $\delta(q_1, 0)= q_2$ *and* $\delta(q_1, 1)= q_4$, add a transition $\delta'(C_{13}, 0)= C_2$ and $\delta'(C_{13}, 1)= C_4$.
- For $\delta(q_2, 0)= q_1$ *and* $\delta(q_2, 1)= q_4$, add a transition $\delta'(C_2, 0)= C_{13}$ and $\delta'(C_2, 1)= C_4$.
- For $\delta(q_4, 0)= \delta(q_4, 1)= q_4$, add a transition $\delta'(C_4, 0)= \delta'(C_4, 1)= C_4$

Example: NFA-to-DFA Conversion in NFA with λ-transition

NFA N



$q_0' = \Lambda\{q_0\} = \Lambda\{1\} = \{1, 3\}$

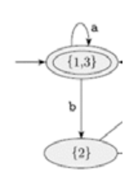
$\forall q' \in Q'$ and $a \in \Sigma$,

$$\delta'(q', a) = \cup_{r \in q', q' \subseteq Q} \Lambda(\delta(r, a))$$

- Add a new start state $\Lambda(\{1\}) = \{1, 3\}$ in M_D.
- For a new {1, 3}, note that

  $\delta(1, a) = \varnothing \Rightarrow \Lambda(\varnothing) = \varnothing$
  $\delta(3, a) = \{1\} \Rightarrow \Lambda(\{1\}) = \{1, 3\}$
  $\Rightarrow \Lambda(\delta(\{1, 3\}\ a)) = \varnothing \cup \{1\} \cup \{1,3\} = \{1, 3\}$.

  $\delta(1, b) = \{2\} \Rightarrow \Lambda(\{2\}) = \{2\}$
  $\delta(3, b) = \varnothing \Rightarrow \Lambda(\varnothing) = \varnothing$
  $\Rightarrow \Lambda(\delta(\{1, 3\}, b)) = \{2\} \cup \varnothing = \{2\}$.

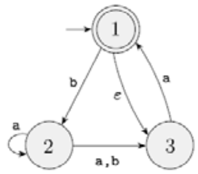


  So, add transitions from {1, 3} to states {1, 3} and {2} for a symbol *a*, *b*, respectively:

  $\delta'(\{1, 3\}, a) = \{1, 3\}$ and $\delta'(\{1, 3\}, b) = \{2\}$.

Example: NFA-to-DFA Conversion in NFA with λ-transition

NFA N



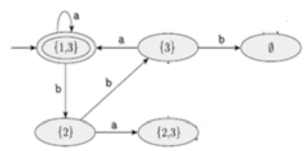
- For a new state {2},

  $\delta(2, a) = \{2, 3\} \Rightarrow \Lambda(\{2, 3\}) = \{2, 3\} \Rightarrow \Lambda(\delta(2, a)) = \{2, 3\}$

  $\delta(2, b) = \{3\} \Rightarrow \Lambda(\{3\}) = \{3\} \Rightarrow \Lambda(\delta(2, b)) = \{3\}$

  So, add transitions from {2} to states {2, 3} and {3} for *a, b*, respectively.

  $\delta'(\{2\}, a) = \{2, 3\}$ and $\delta'(\{2\}, b) = \{3\}$.



- For a new state {3},

  $\delta(3, a) = \{1\} \Rightarrow \Lambda(\{1\}) = \{1, 3\} \Rightarrow \Lambda(\delta(3, a)) = \{1, 3\}$

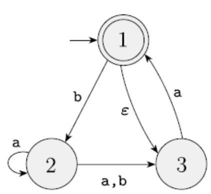
  $\delta(3, b) = \varnothing \Rightarrow \Lambda(\varnothing) = \varnothing \Rightarrow \Lambda(\delta(3, b)) = \varnothing$

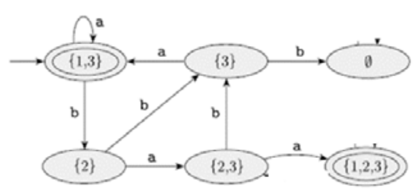
  So, add transitions from {3} to states {1, 3} and $\varnothing$ for *a, b*, respectively.

  $\delta'(\{3\}, a) = \{1, 3\}$ and $\delta'(\{3\}, b) = \varnothing$.

Example: NFA-to-DFA Conversion in NFA with λ-transition

NFA N





- For a new state {2, 3},

  $\delta(2, a) = \{2, 3\} \Rightarrow \Lambda(\{2, 3\}) = \{2, 3\}$

  $\delta(3, a) = \{1\} \Rightarrow \Lambda(\{1\}) = \{1, 3\}$

  $\Rightarrow \Lambda(\delta(\{2, 3\}\ a)) = \{2, 3\} \cup \{1,3\} = \{1, 2, 3\}$.

  $\delta(2, b) = \{3\} \Rightarrow \Lambda(\{3\}) = \{3\}$

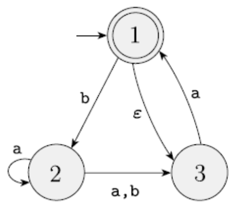
  $\delta(3, b) = \varnothing \Rightarrow \Lambda(\varnothing) = \varnothing \Rightarrow \Lambda(\delta(\{2, 3\}\ b)) = \{3\}$.

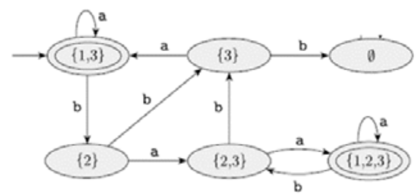
  So, add transitions from {2, 3} to states {1, 2, 3} and {3} for *a, b*, respectively.

  $\delta'(\{2, 3\}, a) = \{1, 2, 3\}$ and $\delta'(\{2, 3\}, b) = \{3\}$.

Example: NFA-to-DFA Conversion in NFA with λ-transition

NFA N





- For a new state {1, 2, 3},

  $\delta(1, a) = \varnothing \rightarrow \Lambda(\varnothing) = \varnothing$,
  $\delta(2, a) = \{2, 3\} \rightarrow \Lambda(\{2, 3\}) = \{2, 3\}$,
  $\delta(3, a) = \{1\} \rightarrow \Lambda(\{1\}) = \{1, 3\}$

  $\Lambda(\delta(\{1, 2, 3\}\ a)) = \{1, 2, 3\}$.

  $\delta(1, b) = \{2\} \rightarrow \Lambda(\{2\}) = \{2\}$,
  $\delta(2, b) = \{3\} \rightarrow \Lambda(\{3\}) = \{3\}$,
  $\delta(3, b) = \varnothing \rightarrow \Lambda(\varnothing) = \varnothing$

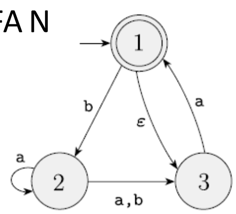
  $\Lambda(\delta(\{1, 2, 3\},\ b)) = \{2, 3\}$

  So, add transitions from {1, 2, 3} to states {1, 2, 3} and {2, 3} for *a, b*, respectively.

  $\delta'(\{1, 2, 3\}, a) = \{1, 2, 3\}$ and $\delta'(\{1, 2, 3\}, b) = \{2, 3\}$.

Example: NFA-to-DFA Conversion in NFA with λ-transition

NFA N



- For a trap state $\varnothing$,

  $\delta(\varnothing, a) = \delta(\varnothing, b) = \varnothing \rightarrow \Lambda(\varnothing) = \varnothing$

  So, add transitions from $\varnothing$ to $\varnothing$ for *a, b*, respectively.

  $\delta'(\varnothing, a) = \delta'(\varnothing, b) = \varnothing$.

The final minimal DFA M: No reduction is necessary.

DFA M