Non-deterministic Finite Automata - examples

$L = \{w \mid \text{the second bit from last is } 1\}$

On input $w = w_1 w_2 \cdots w_n$, an NFA does:

Repeat until end of input:

Current bit is 1: GUESS if it is the second last bit.

guess YES: read the next bit and accept.

guess NO:

Start state: q_0.

Transitions:

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

Accepting state: q_2.

$L = \{w \in \{0,1\}^* \mid \#1(w) \cdot \#0(w) \text{ is even}\}$.

- For $w \in \{0,1\}^*$, $\#1(w) \cdot \#0(w)$ is even iff either (a) $\#1(w)$ is even, or (b) $\#0(w)$ is even.
- From the start state, non-deterministically guess if $\#1(w)$ is even or $\#0(w)$ is even.
  - $\delta(q_s, \epsilon) = \{q_0, q_1\}$.
- If the guess is that $\#1$ is even, then verify that the number of 1's in the input is even and accept if this is the case.
- If the guess is that $\#0$ is even, then verify that the number of 0's in the input is even and accept if this is the case.

Let A and B be two regular languages over $\Sigma = \{0, 1\}$.

$$L = \{w \in \{0, 1\}^* \mid w = x \oplus y \text{ for some } x \in A \text{ and } y \in B \text{ with } |x| = |y|\}$$

(Here $x \oplus y$ is the Boolean Exclusive-OR of the two bit strings x and y.)

Let $M_1 = (Q_1, \{0, 1\}, \delta_1, q_1, F_1)$ be a DFA that recognizes A. Let $M_2 = (Q_2, \{0, 1\}, \delta_2, q_2, F_2)$ be a DFA that recognizes B.

If the current input symbol w_i is a 0 then the current bits $(x\ i, y_i)$ must be either $(0, 0)$ or $(1, 1)$. If the input symbol is a 1 then the two bits whose $\oplus$ is 1 must be either $(0, 1)$ or $(1, 0)$.

Construct an NFA $M = (Q, \{0, 1\}, \delta, q_0, F)$ as follows.

- $Q = Q_1 \times Q_2$.
- For all $p \in Q_1$, $q \in Q_2$, define:
  - $\delta(\langle p, q\rangle, 0) = \{\langle \delta_1(p, 0), \delta_2(q, 0)\rangle, \langle \delta_1(p, 1), \delta_2(q, 1)\rangle\}$.
  - $\delta(\langle p, q\rangle, 1) = \{\langle \delta_1(p, 0), \delta_2(q, 1)\rangle, \langle \delta_1(p, 1), \delta_2(q, 0)\rangle\}$.
- $q_0 = \langle q_1, q_2\rangle$.
- $F = F_1 \times F_2$.

$L = \{w \mid ww \in A, \text{ where } A \text{ is regular}\}$.

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes A.
- Define an NFA $N = (Q', \Sigma, \delta', {q_0}', F')$ as follows.
- N on input w simulates M on ww and accepts w iff M accepts ww.
- Each state of N is a triple of states of M:
  - The first state is the state in which M finds itself on reading the current symbol of w.
  - The last state is the state in which finds itself on reading the current symbol of w having started from the middle state.
  - The middle state is where M finds itself on reading all symbols in w. This is a state that is guessed:
  - The start state is $\{\langle q_0, p, p\rangle \mid \text{for all } p \in Q\}$.
- $F' = \{\langle p, p, r\rangle \mid p \in Q \text{ and } r \in F\}$:
  - On input w, the first state of M reached is the same as the middle state guessed and the final state reached from the guessed middle state is an accept state of M.
- For all $p_1, p_2, p_3 \in Q$ and all $a \in \Sigma$ define $\delta'(\langle p_1, p_2, p_3\rangle, a) = \{\langle \delta(p_1, a), p_2, \delta(p_3, a)\rangle\}$.
  - Run M for one step on input symbol a from the first state p_1 in the triple.
  - Hold the second state p_2 in the triple to check that M reaches this state on reading w.
  - Run M for one step on input symbol a from the third state in the triple.

$L = \{w^R \mid w \in A, \text{ where } A \text{ is regular}\}$.

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes A.
- *Construct* an NFA that, on an input, starts at some accepting state of M and runs M backwards.
- Define an NFA $N = (Q', \Sigma, \delta', {q_0}', F')$ as follows.
- $Q' = Q$.
- $F' = \{q_0\}$.
  - The accept state of the NFA N is the start state of the DFA M.
- For all $p \in Q$ and $a \in \Sigma$ define $\delta'(p, a) = \{q \mid \delta(q, a) = p\}$.
  - From a state p on symbol a, the NFA N picks a state q from the set of all states from which there is a transition to the state p in the DFA M.
- Create a new start state ${q_0}'$ and add an ϵ-transition to each accepting state of M.
  - The NFA N starts its computation from an accept state of the DFA M.

Let A be a regular language.

$$L = \{y \mid \text{there are strings } x, z \text{ such that } |x| = |y| = |z| \text{ and } xyz \in A\}.$$

Let A be recognized by a DFA $M = (Q, \Sigma, \delta, q_0, F)$. An NFA N for the language L, on input y, guesses strings x and z and runs M, in parallel, on the three pieces x , y, and z.

- To run M on x, the machine N guesses the string x one symbol at a time.
- To run M on y, the machine N starts from a guessed state of M. This is the state that M would have reached on x from the start state.
- To run M on z, the machine N guesses the string z one symbol at a time and guesses the state of M from where M has to be run on z. This guessed state is the one that M would have reached on xy from the start state. The state that N reaches from this guessed state on z must be an accept state of M.
- Each state of N is a quintuple $\langle p, q, r, s, t\rangle$ of states of M.
  - p is the state that M is currently in when running on x.
  - q is the state that M is guessed to reach when run on x from the start state.
  - r is the state that M is currently in when it is run from state q on y.
  - s is the state that M is guessed to reach when run on y from state q.
  - t is the state that M is currently in when it is run from state s on z.
- The start state of N is $\langle q_0, p, p, q, q\rangle$, where p and q are guessed states of M.
- An accept state is of the form $\langle p, p, q, q, q_f\rangle$, where q_f is an accept state of M.

The NFA $N = (Q', \Sigma, \delta', q_0', F')$, defined below implements this design:

$$Q' = Q \times Q \times Q \times Q \times Q,$$

$$q_0' = \{\langle q_0, p, p, q, q\rangle | p, q \in Q\},$$

$$\delta'(\langle p, q, r, s, t\rangle, a) = \{\langle \delta(p, b), q, \delta(r, a), s, \delta(t, c)\rangle | b, c, \in \Sigma\},$$

$$F' = \{\langle p, p, q, q, q_f\rangle | p, q \in Q, q_f \in F\}.$$