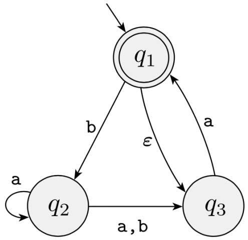
Lec 03. Equivalence of DFA and NFA

Eunjung Kim

Formal definition of NFA

Nondeterministic FA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

- Q a finite set called the <u>states</u>,
- Σ a finite set called the <u>alphabet</u>,
- δ a function from $Q \times \Sigma_\epsilon$ to 2^Q called the <u>transition function</u>,
- $q_0 \in Q$ the <u>start state</u>,
- $F \subseteq Q$ the set of <u>accept states</u>.

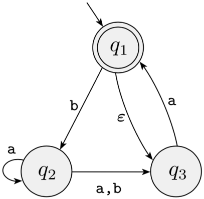


Write a transition table of this NFA.

Language recognized by NFA

NFA N accepts w if

1. w can be written as $y_1, \ldots, y_m$ with $y_i \in \Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ such that there exists a sequence of states $r_0, \ldots, r_m$ satisfying the following:
   - $r_0 = q_0$,
   - $r_{i+1} \quad (??) \quad \delta(q_i, y_i)$,
   - $r_m \in F$.
2. Equivalently, there exists an accepting computation history starting with the (initial) configuration (q_0, w).



Write a computation tree for $w = baabbaa$. How many accepting paths?

Language recognized by NFA

Definition

- Let M be a nondeterministic finite automaton.
- A string $w \in \Sigma^*$ is accepted by M if there exists an accepting computation history.
- $L(M)$ denotes the set of all strings accepted by M.
- A language A is said to be recognized by M if $A = L(M)$.

Equivalence of NFA and DFA

NFA and DFA own the same computational power

For every NFA, there exists a deterministic finite automaton which recognizes the same language (a.k.a. equivalent DFA).

Proof outline.

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
- We want to construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ such that $L(N) = L(M)$.
- Define $Q' := 2^Q$, i.e. the collection of all subsets of Q.
- Let us define δ', $q_0' \in 2^Q$ and $F' \subseteq 2^Q$,

Proof: constructing DFA M, when no ϵ-transition

- $q'_0 = \{q_0\}$.
- transition function δ' from $2^Q \times \Sigma$ to 2^Q:
  for every $R \in 2^Q$ (R is a subset of Q) and every symbol $a \in \Sigma$,
  $$\delta'(R, a) := \bigcup_{r \in R} \delta(r, a)$$
- Define $F' \subseteq 2^Q$ as the collection of all subsets of Q containing at least one accept state of N.

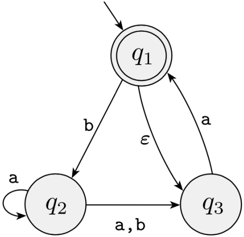
PROOF: CONSTRUCTING DFA M WITH ϵ-TRANSITION

- How to define the <u>initial state</u> for DFA: from a state $q \in Q$ of NFA N, any other state q' that can be reached by reading a string ϵ, can be aggregated with q to form a single state in DFA.
- Define $ext(q) \subseteq Q$ as the set of all states q' of N such that there is a directed path from q to q' in (the state diagram of) N each of whose arcs carries the label ϵ. Extend the definition $ext(X) := \bigcup_{q \in X} ext(q)$.
- transition function δ' from $2^Q \times \Sigma_\epsilon$ to 2^Q:
  for every $R \in 2^Q$ (R is a subset of Q) and every symbol $a \in \Sigma$,

$$\delta'(R, a) := ext(\bigcup_{r \in R} \delta(r, a))$$

- Define $q_0' := ext(q_0) \in 2^Q$. Note that q_0' corresponds to a subset of Q.
- Define $F' \subseteq 2^Q$ as the family of all subsets of Q containing at least one accept state of N.

Constructing DFA M from NFA N, example



PROOF: $L(N) \subseteq L(M)$

Strategy: from an accepting computation history of N on w, build an accepting computation history of M on w.

Let $w = y_1 y_2 \cdots y_s$ for $y_i \in \Sigma$.

- Let $\pi = (q_0, w = w_0), (r_0, w_0), \ldots, (r_i, w_i), \ldots, (r_s, w_s = \epsilon)$ be an accepting computation history of N for w such that

  r_i is reachable from r_{i-1} via a walk (in the transition diagram of N) labelled by $y_i \circ \epsilon^*$.

- Observe: $r_0 \in ext(\{q_0\})$ and $r_i \in ext(\delta(r_{i-1}, y_i))$ for every $i \in [s]$ and $r_s \in F$.

PROOF: $L(N) \subseteq L(M)$

Strategy: from an accepting computation history of N on w, build an accepting computation history of M on w.

Let $w = y_1 y_2 \cdots y_s$ for $y_i \in \Sigma$.

- Let $\pi = (q_0, w = w_0), (r_0, w_0), \ldots, (r_i, w_i), \ldots, (r_s, w_s = \epsilon)$ be an accepting computation history of N for w such that

  r_i is reachable from r_{i-1} via a walk (in the transition diagram of N) labelled by $y_i \circ \epsilon^*$.

- Observe: $r_0 \in ext(\{q_0\})$ and $r_i \in ext(\delta(r_{i-1}, y_i))$ for every $i \in [s]$ and $r_s \in F$.

- Let $Q_0 = q_0'$. Inductively for each $i \in [s-1]$, let

  $$Q_i := \delta'(Q_{i-1}, y_i).$$

- Now we have a computation history for $w = y_1 y_2 \cdots y_s$ in DFA M

  $$\pi' = (Q_0, w_0 = w), (Q_1, w_1), \cdots (Q_t, w_s = \epsilon).$$

PROOF: $L(N) \subseteq L(M)$

Strategy: from an accepting computation history of N on w, build an accepting computation history of M on w.

Let $w = y_1 y_2 \cdots y_s$ for $y_i \in \Sigma$.

- Let $\pi = (q_0, w = w_0), (r_0, w_0), \ldots, (r_i, w_i), \ldots, (r_s, w_s = \epsilon)$ be an accepting computation history of N for w such that

  r_i is reachable from r_{i-1} via a walk (in the transition diagram of N) labelled by $y_i \circ \epsilon^*$.

- Observe: $r_0 \in ext(\{q_0\})$ and $r_i \in ext(\delta(r_{i-1}, y_i))$ for every $i \in [s]$ and $r_s \in F$.
- Let $Q_0 = q_0'$. Inductively for each $i \in [s-1]$, let

$$Q_i := \delta'(Q_{i-1}, y_i).$$

- Now we have a computation history for $w = y_1 y_2 \cdots y_s$ in DFA M

$$\pi' = (Q_0, w_0 = w), (Q_1, w_1), \cdots (Q_t, w_s = \epsilon).$$

Proof: $L(N) \subseteq L(M)$

- It remains to see Q_s is an accept state of M, i.e. the subset $Q_s \subseteq Q$ contains at least one accept state of N.
- This is because $r_0 \in ext(q_0) = Q_0$ and inductively $r_i \in ext(\delta(r_{i-1}, y_i)) \subseteq ext(\delta(Q_{i-1}, y_i) = \delta'(Q_{i-1})$.

Proof: $L(M) \subseteq L(N)$

- Let $\pi' = (R_0, w = w_0), \ldots, (R_i, w_i), \ldots, (R_s, w_s = \epsilon)$ be an accepting computation history of M on w. By definition of computation history $R_i = \delta'(R_{i-1}, y_i)$, where $y_i \in \Sigma$ is the leading symbol of w_{i-1}.
- We construct an accepting computation history of N by following the sequence π' backwardly.
- Observe: for each $q \in R_i \subseteq Q$, there exists a state $q' \in R_{i-1}$ such that from q' to q there is a computation history via a string consisting of y_i followed by ϵ's.
- Now starting from $q_f \in R_s \subseteq Q$, we concatenate computation histories witnessed by the previous observation.

Closure under regular operation

Union operation

Let A_1 and A_2 be two languages recognized by NFA N_1 and N_2 respectively. Then $A_1 \cup A_2$ is recognized by some NFA.

Closure under regular operation

Concatenation operation

Let A_1 and A_2 be two languages recognized by NFA N_1 and N_2 respectively. Then $A_1 \circ A_2$ is recognized by some NFA.

Closure under regular operation

Kleene star operation

Let A a languages recognized by NFA N. Then A^* is recognized by some NFA.

Closure under complementation

Complementation operation

Let A a languages recognized by NFA N. Then $\bar{A}$, that is, $\Sigma^* - A$ is recognized by some NFA.

- For a regular language L, we can obtain a DFA recognizing the complement of L.
- ...using the trick...
- Can we use the same trick for NFA in general?

Closure under intersection

Intersection operation

Let A_1 and A_2 be two languages recognized by NFA N_1 and N_2 respectively. Then $A_1 \cap A_2$ is recognized by some NFA.

- Use the expression that $A_1 \cap A_2 =$??????.
- Combine the above (which ones?) operations on NFAs...
- Direct way with two DFAs M_1 and M_2 by simulating both automata simultaneously.

Closure under intersection

- Direct way with two DFAs M_1 and M_2 by simulating both automata simultaneously.

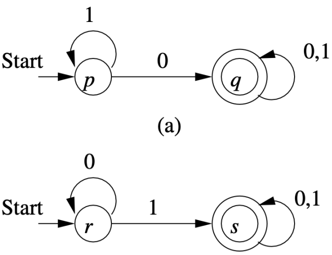


Figure 4.4 (a)-(b), Hopcroft et al. 2014.

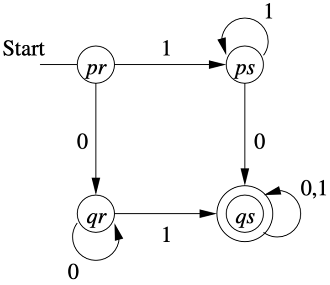


Figure 4.4 (c), Hopcroft et al. 2014.