Regular Languages \(\xrightarrow{}\) DFA Languages

Last lec:

\[ (0^* + 01) \cdot (10)^* \]

NFA
- allows choices
- \(\varepsilon\)-transitions

Ex:

\[ s \xrightarrow{\varepsilon} a_0 \xrightarrow{0} a_1 \xrightarrow{\varepsilon} a_2 \xrightarrow{1} a_3 \]

Then:
If \( L \) is regular,
Then \( L \) is accepted by some NFA \( M \).

Ps:
Let \( r \) be the reg. exp. for \( L \)

\[ r = \{+, \cdot, ^*\} \]

Construct \( M \) recursively

Base case: \( r = \varepsilon, \varepsilon \in \Sigma \)

Induction: (see proof from the last lec)

Plan:

Regular \(\xrightarrow{}\) NFA \(\xrightarrow{\varepsilon}\) DFA

Set \( B \) as Regular lang

\( B \) is accepted by NFA

\( B \) is DFA
* NFA \rightarrow DFA.

Then: It L is accepted by NFA M
Then L \subset \subset some DFA M'

Pf.: Constructive: Power-set Construction.

Let NFA \( M = (Q, \Sigma, s, S, A) \) \( Q \times \{2, 0, 1, 2\} \rightarrow P(Q) \)

Construct DFA \( M' = (Q', \Sigma, s', S', A') \)

\( Q' = P(Q) \)

\( s' = \varepsilon\text{-reach}(s) \)

\( A' = \{ T \in P(Q) \mid T \cap A \neq \emptyset \} \)

\( s': Q' \times \Sigma \rightarrow Q' = P(Q) \)

\( T \in Q', a \in \Sigma \)

\( s'(T, a) = \bigcup_{q \in T} s^*(q, a) \)

Lemma: \( s^*(T, \varepsilon) = \bigcup_{q \in T} s^*(q, \varepsilon) \)

Pf: By induction (base, \ldots)
Then, \( x \in L(M') \iff \delta^*(s', x) \in A' \)

\[ \iff \delta^*(\epsilon, x) \in A' \]

(By Lemma)

\[ \iff \bigcup_{q \in \text{e-read } s_3^*} \delta^*(q, x) \in A' \]

(By def \( \delta^* \) for an NFA)

\[ \iff \delta^*(s, x) \in A' \]

(By def \( A' \))

\[ \iff \delta(s, x) \cap A \neq \emptyset \]

\( \iff x \in L(M) \)

**Ex 1:** Strings ending with 0 or 1.

**NFA**

**DFA**

\[ \delta'(q_0^3, 0) = \delta^*(q_0, 0) = \delta(q_0, 0) = q_1 \]

\[ \delta'(q_0^3, 1) = \delta(q_0, 1) \cup \delta(q_1, 0) = \{q_0, q_3\} \]

\[ \delta'(q_0^3, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_2\} \cup \{q_0, q_3\} = \{q_0, q_2, q_3\} \]

**Ex 2:** \( (0^* (01)^* 1)^* \)
Ex. $2: \ 0^* \ (01)^* \ 1^*$

\[ \delta'(\{q_0, q_1, q_3\}, 0) = \delta(\{q_0, q_1, q_3\}, 0) \cup \delta^*(\{q_3\}) \]
\[ = \{q_0, q_1, q_2, q_3\} \cup \{q_3\} \]
\[ = \{q_0, q_1, q_2, q_3\} \]

DFA $\rightarrow$ Regular Lang.

Thm: If $L$ is accepted by DFA $M$ Then $L$ is regular.

Pf: Constructive. "State Elimination Method"

Intuition:
Pf Sket: Let DFA $M = (Q, \Sigma, \delta, s, F)$

Construct reg. exp. for language $L = L(M)$

By the following method:

1. Introduce $s'$ & $s'$ states
   Add edge from $s'$ to $s$ or $\varepsilon$
   " " " every state in $A$ to $s'$ or $\varepsilon$

2. Eliminate states of $Q$, one-by-one
   Through the following rule:
   $q_i \rightarrow q_k$
   $q_i \rightarrow q_j$
   If may be that $q_j = q_i$.
3. Return label on edge

Ex:

DFA

Req. Ex. 1.

2. \( \Rightarrow \) Remove \( q_2 \)

\( \Rightarrow \) Remove \( q_0 \)

\( \Rightarrow \) Remove \( q_1 \)

\( \Rightarrow \) Remove \( q_{i} \)
Kleene's Theorem (1956):

L is regular iff L is accepted by some DFA M.

Proof: Constructive / Algorithm

\[ \text{Regular} \xrightarrow{\text{Recursive}} \text{NFA} \xrightarrow{\text{Construction}} \text{DFA} \xleftarrow{\text{State Elimination Method}} \]

Cor.: If L is regular then \( \overline{L} \) is regular.

If \( L_1 \), \( L_2 \) are "L1 \& L2" then \( L_1 \cup L_2 \), \( L_1 L_2 \)

Remark: Closed under other ops, like Reverse, Homomorphism, Prefix, Suffix, Subseq, Supseq ...
\[ L = \{ \text{set of all regular languages} \} \]

\[ L \] is "closed under \( \circ \)" if

\[ \forall L, f(L) \in L \]