Last lec: Deterministic Finite Automata

Intuitively: Machines/programs that run in O(1) space & O(n) time.

\[ \Sigma = \{0, 1\} \]

Ex: All strings that have even # 0s

**Def:**

\[ M = (Q, \Sigma, s, A, \delta) \]

\[ \emptyset = \text{set of states} = \{ \text{Even-0}, \text{Odd-0} \} \]

\[ \Sigma = \text{alphabet set} = \{0, 1\} \]

\[ s = \text{start state} = \text{Even-0} \]

\[ A = \text{set of accept states} = \{ \text{Even-0} \} \]

\[ \delta : Q \times \Sigma \to Q \]

\[ \begin{array}{c|ccc}
 & 0 & 1 & \text{I} \\
\hline
\text{E}-0 & 0 & 0 & \text{E}-0 \\
0 & \text{E}-0 & 0 & 0 \\
\end{array} \]

**Def:**

\[ \delta^* : Q \times \Sigma^* \to Q \]

Bare case: \[ \delta^* (q, \varepsilon) = q \]

Induction: \[ \delta^* (q, x) = \delta^* (\delta (q, a), y) \]

\[ x = ay \]

\[ a \in \Sigma, y \in \Sigma^* \]

**Def:**

\[ L(M) = \{ x \in \Sigma^* \mid \delta^* (s, x) \in A \} \]

Ex: All strings that end on 0.
Ex: All strings that end on 01

\[ \delta(q_0, 0110) = q_1 \]
\[ \delta(q_0, 0101) = q_2 \checkmark \]

- \( q_2 \): Last two char are 01
- \( q_1 \): 11 char is 0
- \( q_0 \): None of the above

Ex: (complements) All strings that “do not” end on 01

Construction of DFA for a complement of a language is “easy”

Q: What other operations are easy?

- AND: Any Boolean operation is easy!

Product Construction:

If \( L \) is accepted by DFA \( M \)

Then \( L \) is \( \text{complemented} \) by DFA \( M' \)

Let \( M' = (Q', \Sigma, \delta', s', A) \).
Let $M = (Q, E, s, A, \delta)$. Construct $M' = (Q, E, s, A', \delta')$

$A' = Q \setminus A$

Then:
1. $x \in L(M') \iff \delta'(s, x) \in A'$
2. $\iff \delta'(s, x) \in A$
3. $\iff x \notin L(M) = \emptyset$
4. $\iff x \in \Sigma$

**Ex:** String w/ Even Os

![Graph with states: 0, 1, Ev-0, Ev-1, OD-1, OD-0, M1]

Strings w/ Odd Is

![Graph with states: 0, 1, Ev-1, OD-1, OD-0, M2]

**Q:** Does $x$ have Even Os AND odd Is?

Then: Is $L_1$ accepted by DFA $M_1$ AND $L_2$ accepted by DFA $M_2$?
Then \( L_1 \cap L_2 \) is some DFA \( M \).

\[ M_1 = (Q_1, \Sigma, S_1, A_1, \delta_1) \]
\[ M_2 = (Q_2, \Sigma, S_2, A_2, \delta_2) \]

Construct \( M = (Q, \Sigma, S, A, \delta) \)

\[ Q = Q_1 \times Q_2 \]
\[ S = (S_1, S_2) \]
\[ A = A_1 \times A_2 \]

\[ \delta: Q \times \Sigma \rightarrow Q \]
\[ \delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a)) \]

\[ A \times B = \{ (a, b) \mid a \in A \text{ and } b \in B \} \]
\[ |A \times B| = |A| \times |B| \]

Lemma: \( \forall x \in \Sigma^*, \delta^*(p, q, x) = (\delta_1^*(p, x), \delta_2^*(q, x)) \)

\[ \delta^*(p, q, \epsilon) = (p, q) = (\delta_1^*(p, \epsilon), \delta_2^*(q, \epsilon)) \]

Induction: \( (x+\epsilon) \)

\[ \delta^*(p, q, x) = \delta^*\left(p, q, \frac{a_y}{y} \right) \]
\[ (b \delta^* \delta \bar{b}) = \delta^*\left(\delta(p, a), \frac{a_r}{r} \right), y) \]
\[ = \delta^*\left(\delta_1(p, a), \delta_2(q, a), y \right) \]
\[ (\epsilon, h) = (\delta_1^*(p, a), y), \delta_2^*(q, a), y) \)
\[ (b \delta^* \delta \bar{b}) = (\delta_1^*(p, a), y), \delta_2^*(q, a), y) \)

\[ \delta^*(p, q, x) \]
(by def $M_8$) = \( (\delta_1 (9, q_1), q_2 (9, q_3) ) \)

= \( (\delta_1^* (9, x), \delta_2^* (9, x) ) \)

\[
\begin{align*}
\text{Then,} \\
x \in L(M) & \Rightarrow (\delta^* (s_1, s_2), x) \in A \\
(\text{by above}) & \Rightarrow (\delta_1^* (s_1, x), \delta_2^* (s_2, x)) \in A_1 \times A_2 \\
& \Rightarrow \delta_1^* (s_1, x) \in A_1 \text{ AND } \delta_2^* (s_2, x) \in A_2 \\
& \Rightarrow x \in L(M_1) \text{ AND } x \in L(M_2) \\
& \Rightarrow x \in L_1 \text{ AND } x \in L_2 \\
& \Rightarrow x \in L_1 \cap L_2 \\
\end{align*}
\]

**Ex:** All strings *add* is AND and *or* of.

---

Diagram of finite automata.
Q: \( M_1 \) for \( L_1 \\
M_2 \) for \( L_2 \\
Construct \ M \) for \( L_1 \cup L_2 \)

Everything except accepting states remain as is in the above construction.

\[
L_1 \cup L_2 \quad A = \left\{ (p,q) \in \Sigma_1 \times \Sigma_2 \mid \begin{array}{c}
\text{p} \in A_1 \quad \text{OR} \\
\text{q} \in A_2
\end{array} \right\}
\]

\[
L_1 \cap L_2 \quad A = \left\{ \begin{array}{c}
\text{" "}
\end{array} \right\}
\]

\[
L_1 \setminus L_2 \quad A = \left\{ (p,q) \in \Sigma_1 \times \Sigma_2 \mid \begin{array}{c}
\text{p} \in A_1 \quad \text{AND} \\
\text{q} \notin A_2
\end{array} \right\}
\]

Conclusion: Any boolean ops on languages \( L_1 \) and \( L_2 \) can be constructed corresponding DFAs easily/automatically.

\( \Rightarrow \) Any finite such ops: \( (L_1 \cup L_2) \cup L_3 \)

\( \Rightarrow \) Any finite such ops: \( L_1 \cup L_3 \)