DFA Product Constructions

Wednesday, January 26, 2022 1:52 PM

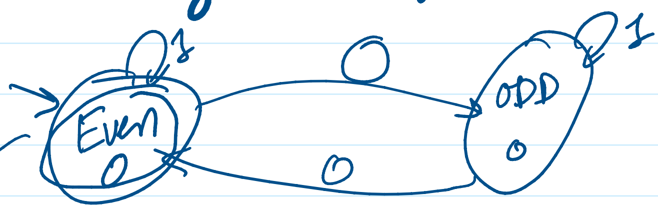
Last lec:

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: $M = (Q, \Sigma, s, A, \delta)$

- Q = set of states = {Even-0, odd-0}
- Σ = alphabet set = $\{0, 1\}$
- s = start state = Even-0
- A = set of accept states = {Even-0}
- $\delta : Q \times \Sigma \to Q$

Q	0	1
E-0	O-0	E-0
O-0	E-0	O-0

Def: $\delta^* : Q \times \Sigma^* \to Q$ (recursively applying δ)

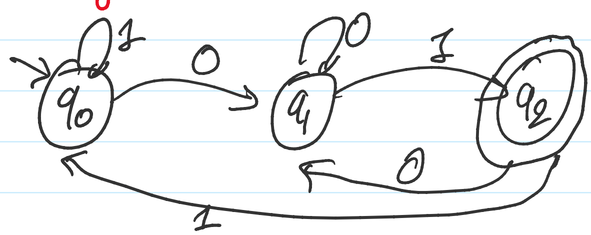
Base case: $\delta^*(q, \varepsilon) = q$

Induction: $\delta^*(q, x) = \delta^*(\delta(q, a), y)$ $\quad x = ay$, $a \in \Sigma$, $y \in \Sigma^*$

Def: $L(M) = \{x \in \Sigma^* \mid \delta^*(s, x) \in A\}$

Ex: All strings that ends at 01

Ex: All strings that ends w/ 01



$\delta^*(q_0, 0110) = q_1$ ✗

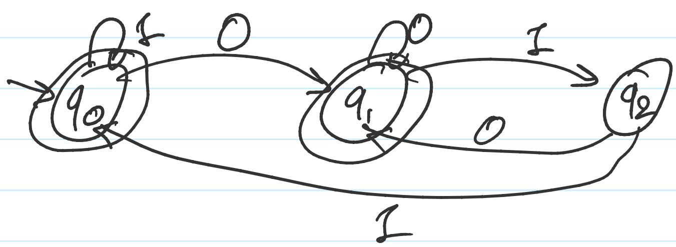
$\delta^*(q_0, 0101) = q_2$ ✓

q_2 = Last two char are 01

q_1 = " char is 0

q_0 = None of the above

Ex: (Complements) All strings that "do not" end w/ 01



Constructing DFA for a complement of a language is "easy"

$\boxed{\text{Q: What other operations are easy?}}$

To Prove:

Ans: Any Boolean operation is easy!

★ Product Construction:

Thm: If L is accepted by DFA M
Then $\bar{L}$ " " " some DFA M'

Pf: Let $M = (Q, \Sigma, s, A, \delta)$.

Pf: Let $M = (Q, \Sigma, s, A, \delta)$.

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

$$A' = Q \setminus A$$

Then:

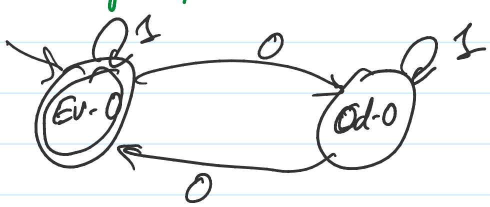
$$x \in L(M') \iff \delta^*(s, x) \in A'$$

$$\iff \delta^*(s, x) \notin A$$

$$\iff x \notin L(M) = L$$

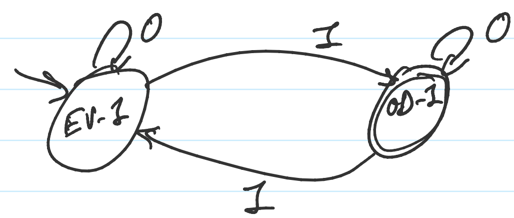
$$\iff x \in \bar{L}$$ ∎

Ex: String w/ Even 0s



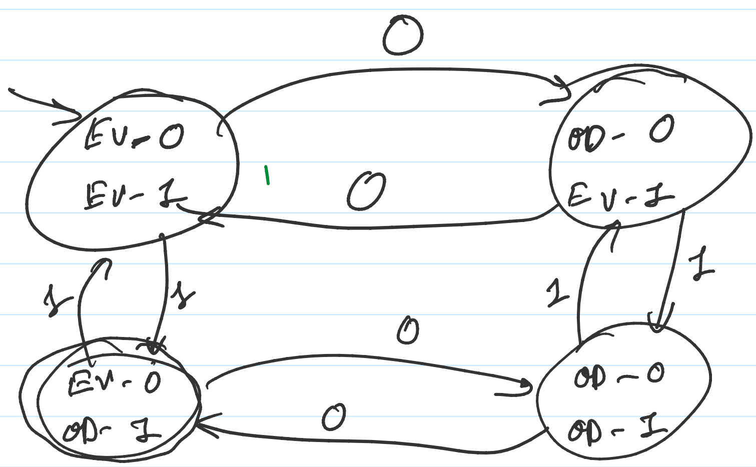
M_1

strings w/ odd 1s



M_2

Q: Does x have Even 0s AND odd 1s?



Thm: If L_1 accepted by DFA M_1 AND
L_2 " " " M_2

Then $L_1 \cap L_2$ " " some DFA M

L_2 " " " M_2 }

Then $L_1 \cap L_2$ " " Some DFA M.

Pf: Intuition: Run M_1 & M_2 in parallel.

Let $M_1 = (Q_1, \Sigma, s_1, A_1, \delta_1)$

$M_2 = (Q_2, \Sigma, s_2, A_2, \delta_2)$

Construct $M = (Q, \Sigma, s, A, \delta)$

$$\left\{\begin{array}{l} Q = Q_1 \times Q_2 \\ s = (s_1, s_2) \\ A = A_1 \times A_2 \\ \delta : Q \times \Sigma \longrightarrow Q \\ \quad \delta((p,q), a) = (\delta_1(p,a), \delta_2(q,a)) \end{array}\right.$$

$A \times B = \{(a,b) \mid a \in A \ \& \ b \in B\}$

$|A \times B| = |A| * |B|$

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

Pf: Base case: $x = \varepsilon$

$$\delta^*((p,q), \varepsilon) = (p,q) = (\delta_1^*(p,\varepsilon), \delta_2^*, (q,\varepsilon))$$

Induction: $(x \neq \varepsilon)$

$x = ay$, $a \in \Sigma, y \in \Sigma^*$

$$\delta^*((p,q), x) = \delta^*((p,q), ay)$$

$$\text{(by } \delta^* \text{ def.)} = \delta^*(\delta((p,q), a), y)$$

$$= \delta^*((\delta_1(p,a), \delta_2(q,a)), y)$$

$$\text{(I.H)} = (\delta_1^*(\delta_1(p,a), y), \delta_2^*(\delta_2(q,a), y))$$

$$\text{(by def of } \delta^*) = (\delta_1^*(p, ay), \delta_2^*(q, ay))$$

(by def 108) $= \left(\delta_1(r, a_?),\ \delta_2(q, a_?)\right)$

$$= \left(\delta_1^*(p, x),\ \delta_2^*(q, x)\right) \quad \blacksquare$$

Then, $x \in L(M) \iff \delta^*((s_1, s_2), x) \in A$

(by above lemma) $\iff \left(\delta_1^*(s_1, x),\ \delta_2^*(s_2, x)\right) \in A_1 \times A_2$

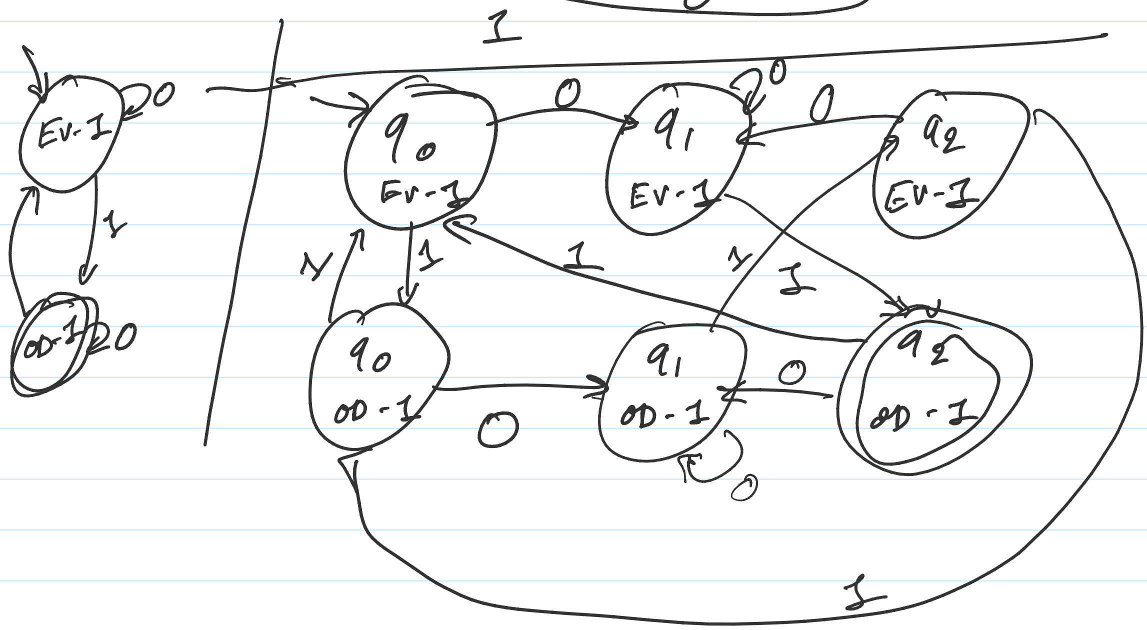
$\iff \delta_1^*(s_1, x) \in A_1$ AND $\delta_2^*(s_2, x) \in A_2$

$\iff x \in L(M_1)$ AND $x \in L(M_2)$

$\iff x \in L_1$ AND $x \in L_2$

$\iff x \in L_1 \cap L_2 \quad \blacksquare$

Ex: All strings $\underbrace{\text{odd \# 1s}}_{M_2}$ AND $\underbrace{\text{end w/ 01}}_{M_1}$.



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

$L_1 \cup L_2 \quad A = \{ (p,q) \in Q_1 \times Q_2 \mid p \in A_1 \text{ OR } q \in A_2 \}$

$L_1 \oplus L_2 \quad A = \{ \quad \text{"} \quad \text{"} \quad \underline{\text{XOR}} \}$

$L_1 \setminus L_2 \quad A = \{ (p,q) \in \underbrace{Q_1 \times Q_2}_{= Q} \mid p \in A_1 \text{ AND } q \notin A_2 \}$

⋮ ⋮

Conclusion: → Any boolean ops on languages & construct corresponding DFAs easily / automatically

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

$\downarrow$
$L' \cup L_3$