CS 4510: Automata and Complexity **Spring 2019**

Lecture 2: Languages and Machines

August 21, 2019

Lecturer: Santosh Vempala *Scribe: Aditi Laddha*

2.1 Definitions

- Alphabet Σ: a finite set of symbols
- Strings : Finite sequence of symbols from Σ
- Σ^*: Set of all strings over Σ
- Language: Set of strings over Γ, $L \subseteq \Gamma$
- Regular language: A language accepted by some DFA.

We say that a DFA M accepts a string w if M stops in a final state on reading w and M recognizes a language L if $L = \{w : M \text{ accepts } w\}$.

Example : $\Sigma = \{0, 1\}$, Σ^* = set of binary strings, $L = \{x \in \Sigma^* : x \text{ has an even number of 1's }\}$.

Given a language L, we can ask the following questions:

- Does there exist a DFA that accepts L?
- Does there exist a Turing Machine that accepts L?

Languages recognized by DFA $\subseteq$ *Languages recognized by TM* $\subseteq$ *All Languages*

Are the set inclusions proper? We will prove that they are in later lectures.

2.2 Composition of Automata

$L = \{x \in \{0, 1\}^* : \text{number of 1's in } x \text{ is not divisible by 2 or 3}\}$. We want to construct a DFA that accepts L.

We can rewrite L as $L = \overline{L}_1 \cap \overline{L}_2$ where $L_1 = \{x \in \{0, 1\}^* : \text{number of 1's in } x \text{ is divisible by 2}\}$ and $L_2 = \{x \in \{0, 1\}^* : \text{number of 1's in } x \text{ is divisible by 3}\}$. In last lecture, we saw how to construct DFAs that accept L_1, L_2.

We can use the DFAs for L_1 and L_2 to construct a DFA that accepts L using the following operations.

2.2.1 Complement

Given a language L recognized by DFA $D = \{Q, \Sigma, \delta, F, q_0\}$, we can construct a DFA D' that recognizes $\overline{L} = \Sigma^* \backslash L$ as $D' = \{Q, \Sigma, \delta, F', q_0\}$ where $F' = Q \backslash F$.

2.2.2 Intersection

Given languages L_1, L_2 that are recognized by DFAs $D_1 = \{Q_1, \Sigma_1, \delta_1, F_1, q_{10}\}, D_2 = \{Q_2, \Sigma_2, \delta_2, F_2, q_{20}\}$, we can construct a DFA D that recognizes $L_1 \cap L_2 = \{w : w \in L_1 \text{ and } w \in L_2\}$ as

$D = \{Q, \Sigma, \delta, F, q_0\}$ where

1. $Q = Q_1 \times Q_2 = \{(q_1, q_2) : q_1 \in Q_1, q_2 \in Q_2\}$
2. $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$
3. $F = F_1 \times F_2$
4. $q_0 = (q_{10}, q_{20})$

Proof of correctness: Let $w = w_1 w_2 \ldots w_n$ and $w \in L_1 \cap L_2$, then there exist 2 sequence of state transitions $q_{11}, q_{12}, \ldots, q_{1n}$ and $q_{21}, q_{22}, \ldots, q_{2n}$ such that $\delta_i(q_{i,j-1}, w_j) = q_{i,j}, \forall i \in \{1, 2\}$ and $\forall j \in \{1, \ldots, n\}$ and $q_{1n} \in F_1, q_{2n} \in F_2$. So, the sequence $(q_{10}, q_{20}), (q_{11}, q_{21}), (q_{12}, q_{22}), \ldots, (q_{1n}, q_{2n})$ forms a valid sequence of state transitions for D on input w(by construction) and $(q_{1n}, q_{2n}) \in F$, so $w \in L_1 \cap L_2 \Rightarrow D$ accepts w.

We can prove the other direction similarly.

Using the complement and intersection constructions, the DFAs for $\overline{L}_1, \overline{L}_2$ and L follow

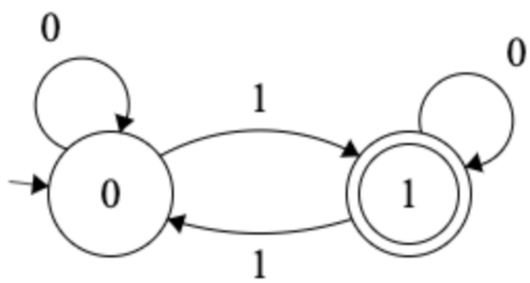

Figure 2.1: A DFA which accepts complement of L_1

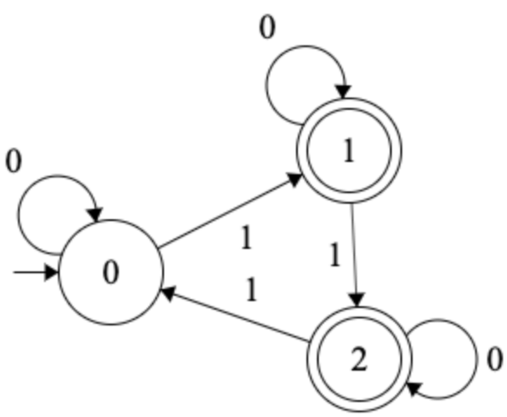

Figure 2.2: A DFA which accepts complement of L_2

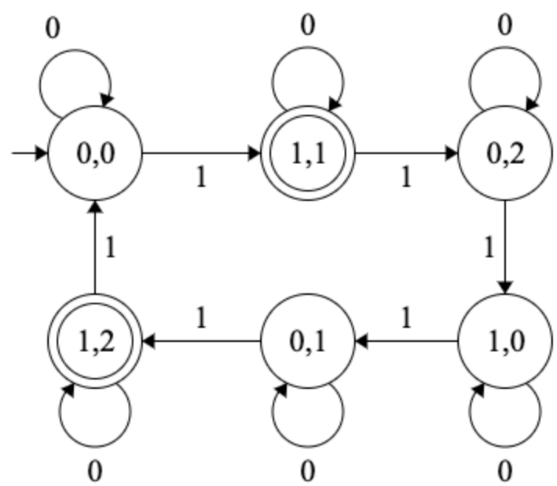

Figure 2.3: A DFA which accepts L

2.2.3 Union

Given languages L_1, L_2 that are recognized by DFAs $D_1 = \{Q_1, \Sigma_1, \delta_1, F_1, q_{10}\}, D_2 = \{Q_2, \Sigma_2, \delta_2, F_2, q_{20}\}$, we can construct a DFA D that recognizes $L_1 \cap L_2 = \{w : w \in L_1 \text{ or } w \in L_2\}$ as
$D = \{Q, \Sigma, \delta, F, q_0\}$ where

1. $Q = Q_1 \times Q_2 = \{(q_1, q_2) : q_1 \in Q_1, q_2 \in Q_2\}$
2. $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$
3. $F = \{F_1 \times Q_2\} \cup \{Q_1 \times F_2\}$
4. $q_0 = (q_{10}, q_{20})$

2.3 Exercises

Construct DFAs or Turing machines that accept the following languages:

1. $\Sigma = \{0, 1\}$, $L = \{a \in \Sigma^* : a\}$ has equal number of 0's and 1's
2. $\Sigma = \{a, \ldots, z\}$, $L = \{a \in \Sigma^* : a\}$ is a palindrome.
3. $\Sigma = \{0, \ldots, 9\}$, $L = \{a \in \Sigma^* : a\}$ is a prime integer

Idea for Turing Machine for problem (1): For input x, start at the leftmost symbol of x and replace it with another symbol α(different from 0,1) and move right searching for the opposite symbol and replace it with α if found. If the end of input is reached without finding it, then reject. Otherwise move to the leftmost non-α letter of x and repeat. If all symbols of x are replaced by α, accept.