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

Lecture 3: TMs and DFAs

August 26, 2019

Lecturer: Santosh Vempala *Scribe: Yanan Wang*

3.1 Turing Machines

3.1.1 Definition

A Turing machine is defined as a tuple $(Q, \Gamma, \sigma, F, q_0)$

- Q: A set of states of finite size
- Γ: tape alphabet, $\Sigma \cup \{\_\}$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, transition function
- $F \subseteq Q$: set of accepting states
- q_0: the initial state

3.1.2 Example

1. Given a language $L = \{0^n1^n\}$, describe a TM that can accept L. (Note: There are no DFAs that can describe L. The proof will be discussed in the next class.)

```
1  if the first symbol is 0 then
2      erase it, scan right and look for the first 1
3      if there is no 1 then
4          reject
5      else
6          mask it
7      go left, stop when meeting the blank and go right by one cell
8      Repeat line 1.
9  if the tape is empty then
10     accept
```

2. Given a language $L = \{a^i b^j c^k | i + j = k\}$, describe a TM that can accept L.

```
1  if the first symbol is a then
2      erase it, scan right and look for the first c
3      if there is no c then
4          reject
5      else
6          mask it
7      go left to the first character
8      repeat line 1
9  else
10     if the first symbol is b then
11         erase it, scan right and find the first c if there is no c then
12             reject
13         else
14             mask it
15         go left to the first character
16         repeat line 10
17 if the tape is empty then
18     accept
```

3.2 DFAs

3.2.1 Definition

A DFA is defined as a tuple $(Q, \Sigma, \sigma, F, q_0)$

- Q: A set of states of finite size
- Σ: input alphabet of finite size
- δ: $Q \times \Sigma \to Q$, transition function
- $F \subseteq Q$: set of accepting states
- q_0: the initial state

3.2.2 Regular languages

A language is regular means that it can be recognized by a finite automaton.

There are two easy but interesting properties of regular languages:

- L is regular $\iff$ $\overline{L}$ is regular. (Proof: Reverse the accepting states, i.e., $F = \overline{F}$.)
- L_1 is regular, L_2 is regular $\implies$ $L_1 \cap L_2$ is regular. (Proof: Refer to lecture note on Aug 21.)
- L_1 is regular, L_2 is regular $\implies$ $L_1 \cup L_2$ is regular.

  Proof: 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 $L_1 \cup L_2 = \{w : w \in L_1 \text{ or } w \in L_2\}$ 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. $F = \{F_1 \times Q_2\} \cup \{Q_1 \times F_2\} = \{(q_1, q_2), q_1 \in F_1 \text{ or } q_2 \in F_2\}$
4. $q_0 = (q_{10}, q_{20})$

Actually, according to De Morgan's laws, $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$. ■

3.3 Nondeterministic Finite Automata

3.3.1 Example

Given a language $L = \{ab | a \in L_1, b \in L_2\}$ where L_1 and L_2 can be described by DFA D_1 and D_2 respectively, describe a finite automaton that can accept L (Note: empty string $\in L_1, L_2$).

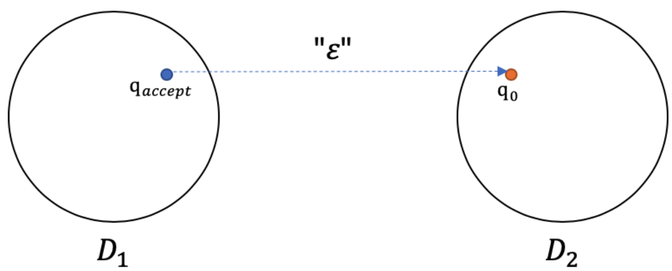

Figure 3.1: A finite automaton that can accept L

ε means that a state can go to another state without reading any symbols.

Due to the existence of ε, the finite automaton becomes nondeterministic, which means the same input can have different sequences of transitions. However, a string is accpeted by such finite automaton iff $\exists$ a valid sequence of transitions ending in an accept state.

3.3.2 Definition

A NFA is defined as a tuple $(Q, \Sigma, \sigma, F, q_0)$

- Q: A set of states of finite size
- Σ: input alphabet of finite size
- δ: $Q \times \Sigma \to Q \cup \{(q, \varepsilon) \to q'\}$, transition function
- $F \subseteq Q$: set of accepting states
- q_0: the initial state

Here are questions:

- Is DFA more powerful than NFA?

  The answer is obviously "NO", since for any DFA, adding a ε transition will get a NFA.

- Is NFA more powerful than DFA? I.e., $\exists$ language L accepted by an NFA but not by any DFAs? The answer will be given in the next class.

Example: An NFA is given as follows:

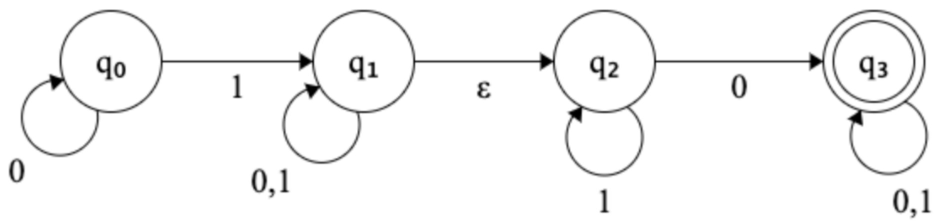

Figure 3.2: An NFA

The regular expression of the NFA is $0^*1\{0,1\}^*0\{0,1\}^*$.
The corresponding DFA is

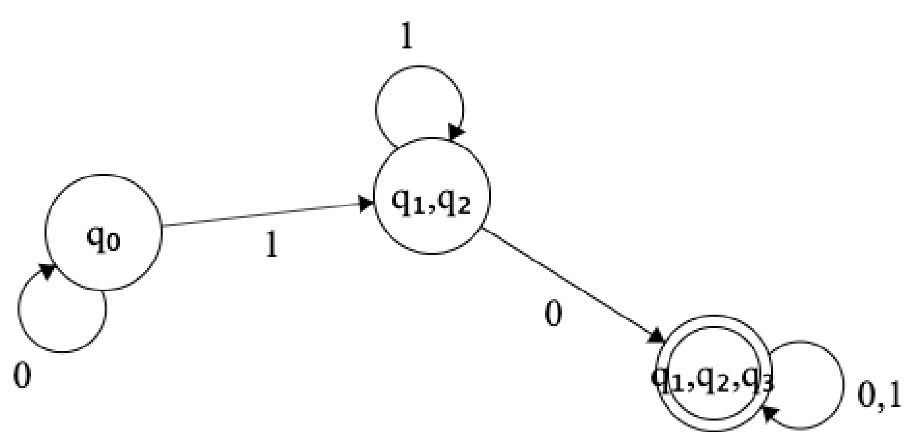

Figure 3.3: A corresponding DFA

3.4 Reference

- Ch 1.1 Finite Automata, ”Introduction to the Theory of Computation”
- Ch 1.2 Nondeterminism, ”Introduction to the Theory of Computation”