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

Lecture 4: Nondeterminism and Regular Languages

August 28, 2019

Lecturer: Santosh Vempala *Scribe: Yanan Wang*

4.1 Nondeterminism

Example: An NFA is given as follows:

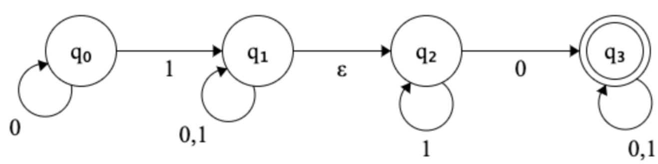

Figure 4.1: An NFA

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

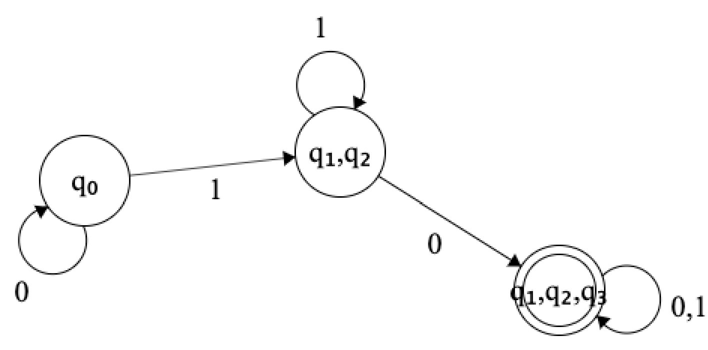

Figure 4.2: A corresponding DFA

Theorem 4.1 *Every NFA can be converted to a DFA.*

Proof: General construction:
Given a NFA with $\{Q, \Sigma, \delta, q_0, F\}$, construct a corresponding DFA $\{Q', \Sigma', \delta', q_0', F'\}$ as follows:
First, we only consider transitions on symbols.

- $Q' = 2^Q$
- For $R \subseteq Q$, i.e., $R \in Q'$, $\delta'(R, a) = \{q | \exists q' \in R, \delta(q', a) = q\}$
- $q_0' = \{q_0\}$
- $F' = \{R \subseteq Q | \exists q \in R \cap F\}$

To add ε transition, we define

$$\varepsilon(R) = \{q | q \in R \text{ or } (\exists q' \in R \text{ and } \delta(q', \delta) = q)\}$$

Then we consider ε transitions and get "ε-extension" of R:

- $Q' = 2^Q$
- For $R \subseteq Q$, i.e., $R \in Q'$, $\delta'(R, a) = \varepsilon(\{q | \exists q' \in R, \delta(q', a) = q\})$
- $q_0' = \varepsilon(\{q_0\})$
- $F' = \varepsilon(\{R \subseteq Q | \exists q \in R \cap F\})$

Hence, every valid sequence of transitions is allowed in the new DFA. All accepting paths remain accepting paths. ■

Corollary 4.2 *L is regular $\iff$ $\exists$ NFA that recognizes it.*

Nondeterminism is **convenient** but not more powerful.
There are some examples showing the convenience of NFA.

- NFA that accepts $L_1 \cup L_2$, where L_1, L_2 are regular languages.

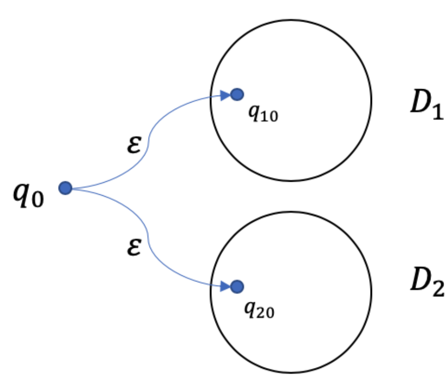

Figure 4.3: NFA that accepts $L_1 \cup L_2$

- NFA that accepts $L_1 \cdot L_2 = \{ab | a \in L_1, b \in L_2\}$, where L_1, L_2 are regular languages.

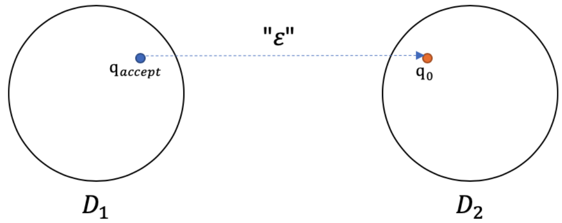

Figure 4.4: NFA that accepts $L_1 \cdot L_2$

- NFA that accepts $L*$, i.e., $L = aa, bc$, $L* = \{\emptyset, aa, bc, aaaa, bcbc, aabc, \cdots\}$.

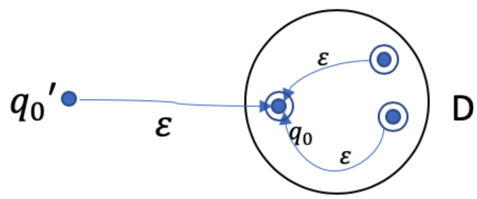

Figure 4.5: NFA that accepts $L*$

4.2 Regular expressions

4.2.1 Definition

Say that R is a **regular expression** if R is

- $R = \emptyset$, empty language
- $R = a, \forall a \in \Sigma$
- $R = \{\varepsilon\}$, language with empty string
- $R = (R_1 \cup R_2)$, where R_1 and R_2 are regular expressions
- $R = R_1 \cdot R_2$, where R_1 and R_2 are regular expressions
- $R = R_1^*$, where R_1 is regular expression

Example: Describe a regular expression for binary strings with an even # of 1's.
Solution: $(0^*10^*1)^*0^*$.

Theorem 4.3 *L is regular $\iff$ $\exists$ regular expression for L.*

Proof:
First, we prove that for any regular expression R that represents language L, L is regular. According to 4.2, we just have to show that there is a NFA/DFA that recognizes L.
Let's convert R into an NFA N. We consider the six cases in the formal definition of regular expressions.

1. $R = a, \forall a \in \Sigma$. Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$.

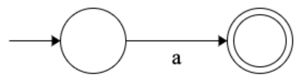

Figure 4.6: NFA that recognizes $L(R) = a$

2. $R = \varepsilon$. Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.

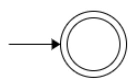

Figure 4.7: NFA that recognizes $L(R) = \{\varepsilon\}$

3. $R = \emptyset$. Then $L(R) = \emptyset$, and the following NFA recognizes $L(R)$.

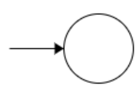

Figure 4.8: NFA that recognizes $L(R) = \emptyset$

4. $R = R_1 \cup R_2$. Refer to the previous lecture note.
5. $R = R_1 \cdot R_2$. Refer to the previous lecture note.
6. $R = R_1^*$. Refer to the previous lecture note.

The other direction is also elementary but more tedious, please refer to the book. ■

4.3 Nondeterministic TMs

4.3.1 Definition

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

- Q: A set of states of finite size
- Γ: tape alphabet, $\Sigma \cup \{\_\}$
- δ: list of allowed transitions and can be more than 1 for (q, a)
- $F \subseteq Q$: set of accepting states
- q_0: the initial state

NTM accepts an input iff $\exists$ valid computation path leading to an accept state.
Question: Is NTM more powerful than TM?
Let's take a look at some examples.

1. Does there exist a path from s to t in graph $G = (V, E)$?

   Deterministic TM: BFS, DFA
   Nondeterministic TM: Guess next vertex! (All edges are valid transitions)

2. Does there exist a Hamilton cycle (visiting all vertices once) in G?

3. Does there exist a TSP of length $\leq L$, given a graph (map) with edge lengths?

Note: Languages recognized by TMs are called "Recursively Enumerable".

4.4 Reference

- Ch 1.2 Nondeterminism, "Introduction to the Theory of Computation"
- Ch 1.3 Regular Expressions, "Introduction to the Theory of Computation"
- Ch 3.2 Variants of Turing Machines, "Introduction to the Theory of Computation"