CS:4330 Theory of Computation

Spring 2018

Regular Languages

Nondeterminism

Haniel Barbosa

THE UNIVERSITY OF IOWA

Readings for this lecture

Chapter 1 of [Sipser 1996], 3rd edition. Sections 1.2.

Nondeterminism

- So far in our discussion, every step of a finite automaton computation follows a unique way from the preceding step.

- We call this a *deterministic computation*. In a *nondeterministic computation*, choices may exist for the next state at any point

- Nondeterminism is a generalization of determinism; hence, every finite automaton is a nondeterministic finite automaton (NFA)

Two ways to introduce Nondeterminism

- More choices for the next state: zero, one, or many arrows may exist from each state

- A state may change to the next state without spending an input symbol: ϵ-transitions

Formal Definition of NFA

- An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ in which:
  - Q is a finite set called the *states*
  - Σ is a finite set called the *alphabet*
  - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathscr{P}(Q)$ is the transition function, in which $\mathscr{P}(Q)$ is the power set of Q
  - $q_0 \in Q$ is the *start state*, also called *initial stat*
  - $F \subseteq Q$ is the set of *accepted states*, also called the *final states*

- In a DFA, the transition function is $\delta : Q \times \Sigma \to Q$

- **Notation:** For any alphabet Σ,

$$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$$

Tree Computation of NFA

A way to think of a nondeterministic computation is as a tree of possibilities

- ▷ The root of the tree corresponds to the start of the computation
- ▷ Every branching point in the tree corresponds to a point in the computation at which the machine has multiple choices
- ▷ The machine accepts if at least one of the computation branches ends in a an accept state

Computation by an NFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w be a string over Σ. We say that N accepts w if $w = \mathrm{a}_0 \ldots \mathrm{a}_n$, $\mathrm{a_i} \in \Sigma_\epsilon$, $0 \leq i \leq n$, and a sequence of states $r_0, \ldots, r_n$ exists in Q such that:

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, \mathrm{a}_{i+1})$ for $i = 0, 1, \ldots, n-1$
3. $r_n \in F$

Condition (1) says where the machine starts

Condition (2) says that state r_{i+1} is one of the allowable new states when N is in state r_i and reads a_{i+1}. Note that $\delta(r_i, \mathrm{a}_{i+1})$ is set of states

Condition (3) says when the machine accepts the input if the last state is in the accept state set

Why NFA?

- ▷ Constructing NFA is sometimes easier than constructing DFA
- ▷ An NFA may be much smaller than a DFA that performs the same task
- ▷ Computation of NFA is usually more expensive than that of DFA
- ▷ Every NFA can be converted into an equivalent DFA
- ▷ NFA provides good introduction to nondeterminism in more powerful computational models

Equivalence of NFA and DFA

Theorem

Every NFA has an equivalent DFA to recognize the same language

- ▷ DFAs and NFAs recognize the same class of languages

- ▷ This equivalence is both surprising and useful
  - ▶ It is surprising because NFAs appears to have more power than DFAs, so we might expect that NFA recognizes more languages
  - ▶ It is useful because describing an NFA for a given language sometimes is much easier than describing a DFA

Equivalence of NFA and DFA

Theorem

Every NFA has an equivalent DFA to recognize the same language

Proof idea: Build an equivalent DFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing the language A. We construct the DFA M recognizing A

- ▷ Before doing the full construction, consider first the easier case when N has no ϵ transitions

Equivalence of NFA and DFA

Theorem

Every NFA has an equivalent DFA to recognize the same language

Proof idea: Build an equivalent DFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing the language A. We construct the DFA M recognizing A

- ▷ Before doing the full construction, consider first the easier case when N has no ϵ transitions
- ▷ Then we consider the ϵ transitions
- ▷ **Notation:** For any $R \subseteq Q$ define $E(R)$ to be the collection of states that can be reached by R by going only along ϵ transitions, including the members of R themselves. Formally:

$$E(R) = R \cup \{q \in Q \mid \exists r_1 \in R, r_2, \ldots, r_k \in Q, r_{i+1} \in \delta(r_i, \epsilon), r_k = q\}$$