Lecture 2: Regular Expression

Instructor: Ketan Mulmuley
Scriber: Yuan Li

January 8, 2015

In the last lecture, we proved that DFA, NFA, and NFA with ϵ-moves are equivalent. Recall that a language $L \subseteq \Sigma^*$ is called regular if $L = L(M)$ for some DFA M (or equivalently NFA, or NFA with ϵ-moves).

In this lecture, we will give a syntactic characterization of regular languages.

1 Regular Expression

For example, $R = (0+1)^*00(0+1)^*$ is a regular expression representing all binary strings containing two consecutive zeros. Here, $+$ means or, $*$ means repetition any number of times (including zero times).

Formally, regular expression is defined by the following rules.

- ϕ is a regular expression, which denotes $\{\} \subseteq \Sigma^*$.
- ϵ is a regular expression, which denotes $\{\epsilon\}$.
- For each $a \in \Sigma$, a is a regular expression, which denotes $\{a\}$.
- If r and s are regular expressions, $r+s$, rs, r^* are also regular expressions. If r, s denote $R, S \subseteq \Sigma^*$ respectively, then $r+s$ denotes $R \cup S$, rs denotes $RS = \{uv : u \in R, v \in S\}$, and r^* denotes $R^* = \bigcup_{i \geq 0} R^i$, where $R^0 = \{\epsilon\}$ and $R^i = \underbrace{RR \cdots R}_{i \text{ times}}$.

The next theorem says a language is regular if and only if it can be described by regular expression.

Theorem 1.1. *Regular expression is equivalent to NFA with ϵ-moves (and thus equivalent to DFA, NFA).*

Proof. (Regular expression $\Rightarrow$ NFA with ϵ-moves) We will prove, if L is accepted by a regular expression, then there exists an NFA with ϵ-moves M such that $L = L(M)$.

Basis: if $r = \emptyset$, let M be an NFA with only initial state (no final state); if $r = \epsilon$, let M be an NFA with one state, which is both the initial state and final state. If $r = a$, let M be an NFA with one initial state, and one final state, connected by an arrow a.

For induction step, let r and s are two regular expressions equivalent to M_1, M_2 respectively, and assume both r and s have at most one final state. Expression $r + s$ is equivalent to M defined as follows.

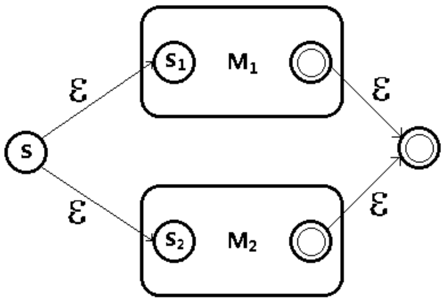


Figure 1: ϵ-NFA accepting expression $r + s$

Expression rs is equivalent to M defined as follows.

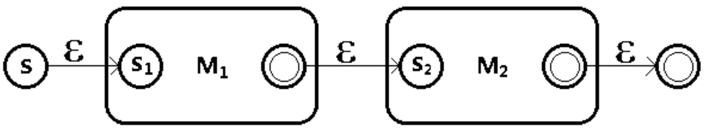


Figure 2: ϵ-NFA accepting expression rs

Expression r^* is equivalent to M defined as follows.

Since in the last class, we see that NFA with ϵ-moves is equivalent to DFA. For the other direction, we need to prove DFA $\Rightarrow$ regular expression,

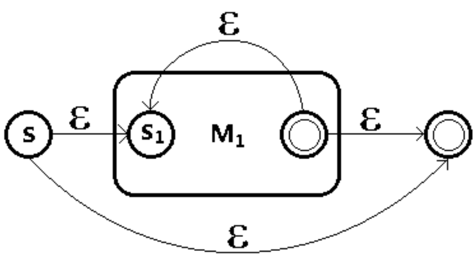


Figure 3: ϵ-NFA accepting expression r^*

that is, given any DFA M, there exists a regular expression r such that $L(M)$ is denoted by r.

The idea is to define recursively like dynamic programming. Let $M = (Q, \Sigma, \delta, q, F \subseteq Q)$, where $Q = \{q_1, q_2, \ldots, q_n\}$. We will recursively define regular expression

$$R_{ij}^k, 1 \leq i, j, k \leq n$$

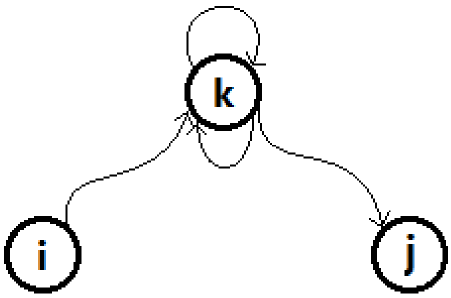
and i, j could be greater than k. **Let $L(R_{ij}^k)$ be the set of all strings in Σ that take M from state i to state j without going through (enter and leave) any state numbered higher than k.**

By definition, R_{ij}^n is the set of all strings that take M from state i to state j. Thus, $L(M) = \bigcup_{j \in F} L(R_{1j}^n)$. Let $R = \bigoplus_{j \in F} R_{1j}^n$, and we have $L(R) = L(M)$.

Do induction on k. Let

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}.$$

In words, consider a path from state i to state j without going through any state numbered higher than k. There are two possibilities — either the path does *not* going through any state numbered higher than $k-1$, or the path first goes from i to k (without going through any state numbered higher than $k-1$), then loops at state k (without going through any state numbered higher than $k-1$), and finally goes from state k to state j (without going through any state numbered higher than $k-1$).



For the induction basis, i.e., $k = 0$, let

$$R_{ij}^0 = \begin{cases} a_1 + \ldots + a_s, & \text{if } i \neq j, \\ a_1 + \ldots + a_s + \epsilon, & \text{if } i = j, \end{cases}$$

where $a_1, \ldots, a_s$ are the labels of the arrows from i to j. $\square$

Note that in above construction, the length of the regular expression could be exponential in n.

2 Two-way Finite Automaton

A two-way finite automaton is a finite automaton where its head can move in both directions. It accepts the input string if it moves its head to the right end and enter a final state at the same time.

Formally, a two-way deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, where $\delta : Q \times \Sigma \rightarrow Q \times \{L, R\}$.

In order to formally define the language accepted by M, let us define *instantaneous description* (ID) first. An instantaneous description is a snapshot of runtime DFA. An ID of a two-way finite automaton is a string wqx, where $w, q \in \Sigma^*$ and $q \in Q$, which means the current state is q, and the head is on the first symbol of x.

Define a relation $\mapsto_M$, where $I \mapsto_M J$ means that J is the next ID of I. Specifically, if $I = a_1 \cdots a_{i-1} q a_i \cdots a_n$, $i > 1$, then

- $J = a_1 \cdots a_i p a_{i+1} \cdots a_n$ if $\delta(q, a_i) = (p, R)$.
- $J = a_1 \cdots a_{i-2} p a_{i-1} \cdots a_n$ if $\delta(q, a_i) = (p, L)$.

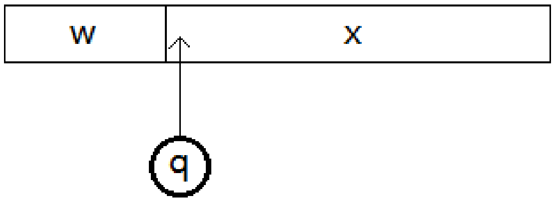


Figure 4: Instantaneous Description of 2-DFA

If $i = 1$ and $\delta(q, a_1) = (p, L)$, then the string is rejected.

Let $\mapsto_M^*$ denote the reflexive transitive closure of $\mapsto_M$, that is, $I \mapsto_M^* J$ if J can be reached from I by applying $\mapsto_M$ any number of times (including zero times). Let $w \in \Sigma^*$ be the input string, then

$$L(M) = \{w \in \Sigma^* : q_0 w \mapsto_M^* wp \text{ for some } p \in F\}.$$

Is two-way DFA strictly stronger than DFA? The answer is not so obvious. In the next lecture, we will prove that two-way DFA and DFA are in fact equivalent.