Finite Automata and Regular Expression

Theorems - Unit II

SUSAN ELIAS

Professor
Department of Computer Science & Engineering
Sri Venkateswara College of Engineering

September 17, 2012

Introduction

Chapter - 3

Regular Expressions and Languages

Chapter - 4

Properties of Regular Languages

Unit II - Guidelines

Formal Definitions

- Regular Expressions
- Algebraic Laws for Regular Expressions

Theorems

- DFA to Regular Expression - Theorem 3.4
- Convert Regular Expression to Automata - Theorem 3.7
- Pumping Lemma - Theorem 4.1
- Closure Property of Regular Languages

Problems

Unit II - Guidelines

Formal Definitions

Theorems

Problems

- Given a Language write the Regular Expression
- Convert the given DFA to Regular Expression - R_{ij} method
- Convert the given DFA to Regular Expression - State Elimination
- Convert Regular Expression to Automata - Problems
- Pumping Lemma - Problems
- Equivalence and Minimisation

DFA to Regular Expressions - Theorem 3.4

Theorem 3.4:
If L = L(A) for some DFA A, then there is a regular expression R such that L = L(R)
Proof:

- Let us suppose that A's states are {1,2,....n} for some integer *n*.
- Let us use R_{ij}^{k} as the name of a regular expression whose language is the set of strings w such that w is the label of a path from state i to state j in A and that path has no intermediate node whose number is greater than k.
- To construct the expression R_{ij}^{k} an inductive definition is used starting at $k = 0$ and finally reaching $k = n$

DFA to Regular Expressions - Theorem 3.4

Basis: The basis is *k=0* i.e paths having no intermediate states. Two kinds of paths meet this condition.

- An arc from state i to state j
- A path of length 0 that consists only of some node i.

The regular expressions generated in the Basis step can be

- If there is no arc $R_{ij}^{0} = \phi$
- If $i = j$ then $R_{ij}^{0} = \epsilon$
- If there is exactly one symbol a then $R_{ij}^{0} = a$
- If there are symbols $a_1, a_2, ..., a_3$ then $R_{ij}^{0} = a_1 + a_2 + ... + a_k$

DFA to Regular Expressions - Theorem 3.4

Induction: Suppose there is a path from state i to state j that goes through no state higher than k there are two possible cases to consider.

- The path does not go through state k at all. In this case the label of the path is in the language of R_{ij}^{k-1}
- The path goes through state k at least once. Draw the figure 3.3. The set of labels for all paths of this type is represented by the regular expression $R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$

Combining the expressions for the paths of the two types we get

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

The regular expression for the language of the automaton is then the sum (union) of all expressions R_{1j}^{n} such that state j is an accepting state.

Regular Expressions to Automata - Theorem 3.7

Theorem 3.7:
Every language defined by a regular expression is also defined by a finite automata

Proof: Suppose $L = L(R)$ for a regular expression R. We show that $L = L(E)$ for some $\epsilon - NFA$ E with

1. Exactly one accepting state
2. No arcs into the initial state
3. No arcs out of the accepting state

The proof is by structural induction on R, following the recursive definition of regular expressions.

Regular Expressions to Automata - Theorem 3.7

Basis: There are three parts to the basis. Draw diagrams to represent $\{\epsilon\}$, ϕ and $\{a\}$. It is easy to check from the diagrams that all three of them satisfy conditions (1), (2) and (3) of the inductive hypothesis.

Induction: Draw diagrams for $R + S$, RS and R^*

- Assuming that the induction hypothesis is true for smaller expressions R and S we can observe that the language of the automaton in $L(R) \cup L(S)$ in case (1)
- Assuming that the induction hypothesis is true for smaller expressions R and S we can observe that the language of the automaton in $L(R)L(S)$ in case (2)
- Assuming that the induction hypothesis is true for smaller expression R we can observe that the language of the automaton is $\{\epsilon\}$, $L(R)$, $L(R)L(R)$, $L(R)L(R)L(R)$ in case (3)

The constructed automata satisfy the three conditions of the inductive hypothesis - one accepting state, with no arcs into the initial state or out of the accepting state

Pumping Lemma for Regular Languages - Theorem 4.1

Theorem 4.1: Let L be a regular language. Then there exists a constant n (which depends on L) such that for every string w in L such that $|w| \geq n$, we can break w into three strings, $w = xyz$, such that:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. For all $k \geq 0$, the string xy^kz is also in L

Proof:

- Suppose L is regular, then $L = L(A)$ for some DFA A.
- Suppose A has n states. Now consider any string w of length n or more, say $w = a_1a_2...a_m$, where $m \geq n$ and each a_i is an input symbol.
- For $i = 0, 1, \cdots, n$ define p_i to be the state in which A is in after reading the first i symbols of w. Note that $p_0 = q_0$

Pumping Lemma for Regular Languages - Theorem 4.1

It is not possible for $n+1$ different p_i's for $i = 0, 1, \cdots, n$ to be distinct, since there are only n different states. Thus we can find two different integers i and j with $0 \leq i \leq j \leq n$ such that $p_i = p_j$. Now we can break $w = xyz$ as follows:

1. $x = a_1 a_2 \ldots a_i$
2. $y = a_{i+1} a_{i+2} \ldots a_j$
3. $z = a_{j+1} a_{j+2} \ldots a_m$

Draw the Figure 4.1

- x takes us to p_i once (x may be empty in that case $i = 0$)
- y takes us from p_i back to p_i (y cannot be empty as i is strictly less than j)
- z is the balance of w (z may be empty if $j = n = m$)

For the input $xy^k w$ explain the figure for $k = 0$ and for $k \geq 0$