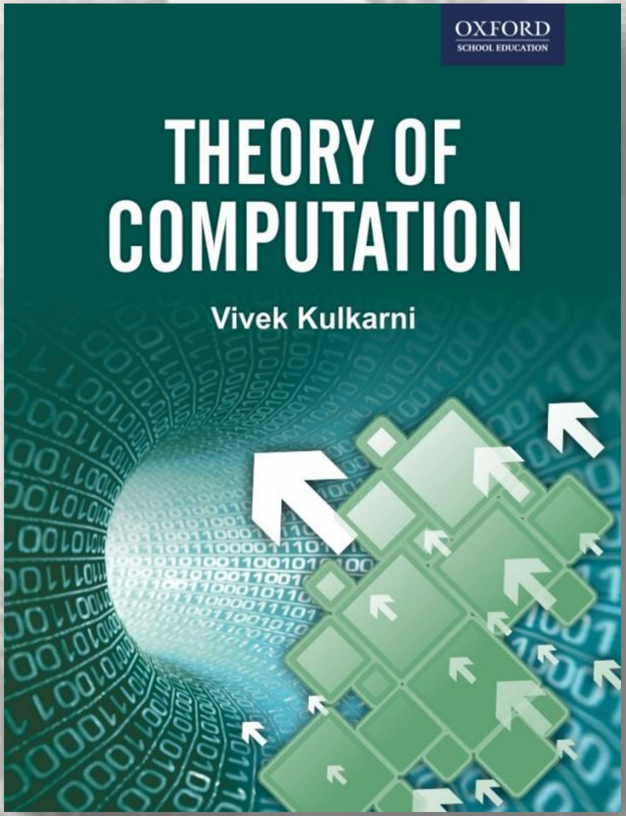
OXFORD
HIGHER EDUCATION

THEORY OF COMPUTATION

Vivek Kulkarni

Slides for Faculty Assistance



Chapter 3

Regular Expressions

Author: Vivek Kulkarni

vivek_Kulkarni@yahoo.com

Outline

- Following topics are covered in the slides:
  - The concept of regular expressions (RE)
  - Equivalence of regular expressions and finite automata
  - Construction of a DFA from the given regular expression
  - Obtaining a regular expression for the language accepted by a DFA
  - Arden's theorem
  - Closure properties of regular languages (or sets)
  - Pumping lemma for regular languages
  - Applications of regular expressions / finite automata

Regular Expression (RE)

- The languages accepted by finite automata (FA) are described or represented by simple expressions called *regular expressions* (RE). Regular expressions, say r are like the short-form notations that denote regular languages (or regular sets) say, $L(r)$.

- The class of regular expressions over Σ is defined recursively as follows:
  - 1. Regular expressions over Σ, include letters, ϕ (empty set), and ϵ (empty string of length zero).
  - 2. Every symbol $a \in \Sigma$ is a regular expression over Σ.
  - 3. If $R1$ and $R2$ are regular expressions over Σ, then so are $(R1 + R2)$, $(R1 \cdot R2)$, and $(R1)^*$, where '+' indicates *alternation* (parallel path), the operation '.' denotes *concatenation* (series connection), and '*' denotes *iteration* (*closure* or repetitive concatenation).
  - 4. Regular expressions are only those that are obtained using rules 1–3.

Regular Expression Examples

- Language consisting of all strings over $\Sigma = \{0, 1\}$ with at least two consecutive 0's can be denoted using RE as,

$$r = (0 + 1)^* \cdot 0 \cdot 0 \cdot (0 + 1)^*$$

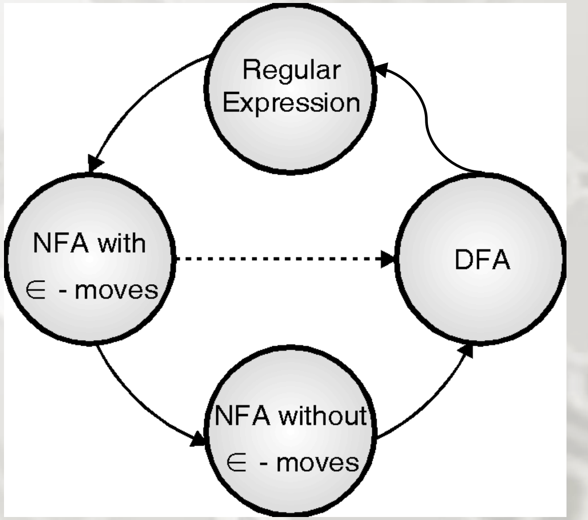
- If $L(r)$ = set of all strings over $\Sigma = \{0, 1, 2\}$, such that at least one 0 is followed by at least one 1, which is followed by at least one 2 then, $r = 0 \cdot 0^* \cdot 1 \cdot 1^* \cdot 2 \cdot 2^*$ or, $r = 0^+ \cdot 1^+ \cdot 2^+$

- The language over $\Sigma = \{0, 1\}$ containing all possible combinations of 0's and 1's but not having two consecutive '0's can be represented using RE as,

$$r = (0 + \in) \cdot (1 + 10)^*$$

Equivalence of Regular Expressions and Finite Automata

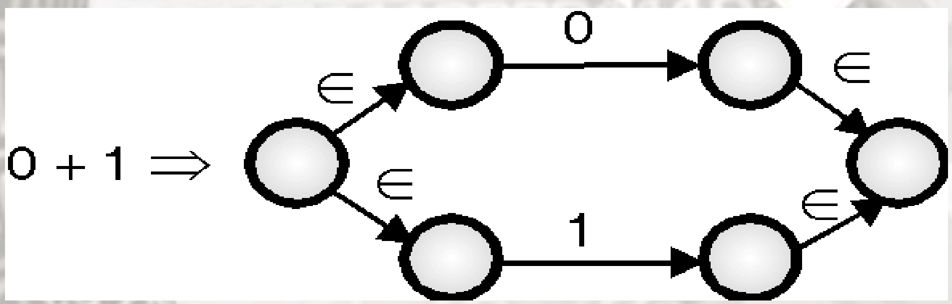
- **Kleene's theorem** is stated in two parts:
  - Any regular language is accepted by a finite automaton.
  - Languages accepted by FA are regular.



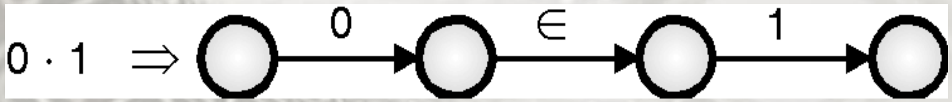
- Given a regular expression one can obtain an equivalent DFA accepting the language represented by the regular expression. The converse is true as well.

Regular Expression to NFA with ∈ -moves Conversion

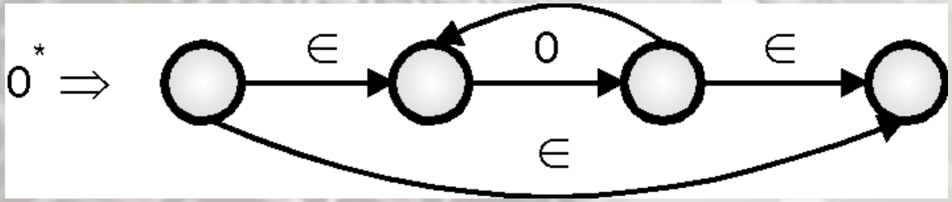
Rules for constructing NFA with ∈ -moves from given regular expression



Parallel paths

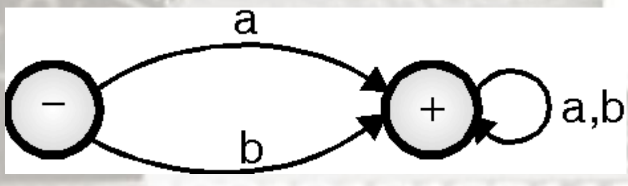


Series

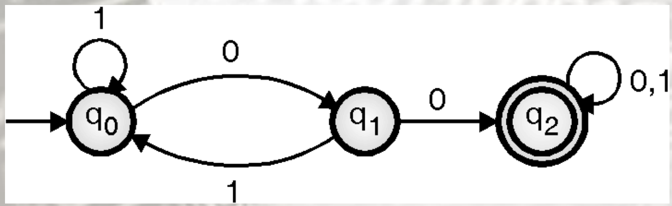


Closure

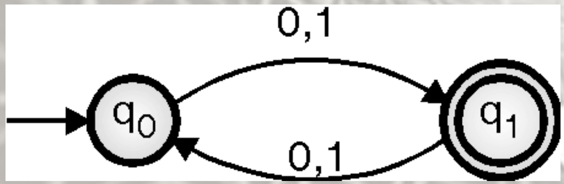
DFA to RE - Examples



$r = (a + b) \cdot (a + b)^*$



$r = 1^* 0 0 (0 + 1)^* + (1^* 01)^* 00 (0 + 1)^*$



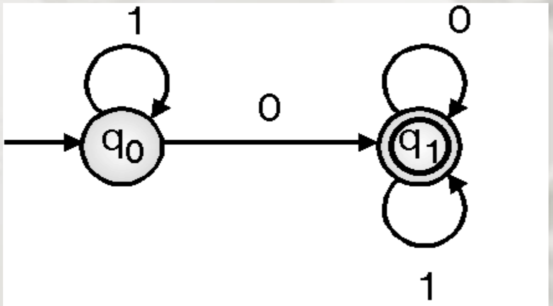
$r = (0 + 1) \cdot [(0 + 1) \cdot (0 + 1)]^*$

Iterative method for obtaining RE from DFA

- Let us consider any general DFA M, with Q 5 {1, 2, 3, 4, …, n}.
- Let us also use the label $R^{(k)}_{ij}$, which represents a regular expression, and whose language is the set of all strings w such that there is a path w available from state i to state j in the transition graph for M. The only restriction here is that the path does not traverse through any state, whose number is greater than k.
- The expression $R^{(k)}_{ij}$, when built through inductive definition, where we start with k = 0 and incrementally build the expression till $k = n$. In this way, we achieve all possible paths from i to j that traverse through all the possible states available in M.
- For k = 0 there will be no intermediate state. Hence, for k = 0, we rely only on the direct transitions that are available.
- When i and j are initial and final states respectively and if $k = n$, then $R^{(k)}_{ij}$ represents the RE equivalent to the DFA M.

Arden's Theorem

- If 'P', 'Q' and 'R' are regular expressions and,
  (i) If R = P + RQ or R = RQ + P then, R can be simplified as, R = PQ*
  (ii) If R = P + QR or, R = QR + P then, R can be simplified as, R = Q* P



- For the example DFA,

$$q_0 = q_0 1 + \in = \in 1^* = 1^*$$

$$q_1 = q_0 0 + q_1 (0 + 1) = 1^* 0 + q_1 (0 + 1) = 1^* 0 (0 + 1)^*$$

As q_1 is the final state for the DFA, RE equivalent to the DFA is,

$$r = 1^* 0 (0 + 1)^*$$

Closure Properties of Regular Sets

- Regular languages – languages represented by regular expressions are termed as Regular Sets.
- Regular sets (or, regular languages) are closed under the operations –
  - **Union**: '$R_1 + R_2$' denotes all the strings that are either denoted by 'R_1' or 'R_2'. Thus, $L(R_1 + R_2) = L(R_1) \cup L(R_2)$
  - **Concatenation**: '$R_1 \cdot R_2$' denotes all the strings that are denoted by 'R_1' concatenated with the strings denoted by 'R_2'. Thus, $L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2)$.
  - **kleene closure**: 'R_1*' denotes all the strings that are denoted by 'R' concatenated to itself zero or more number of times. Thus, $L(R^*) = L(R) \cdot L(R) \cdot L(R) \cdot L(R)$ …… zero or more number of times.

Pumping Lemma for Regular Languages

- It states that given any sufficiently long string accepted by an FSM, we can find a substring near the beginning of the string that may be repeated (or pumped) as many times as we like and the resulting string will still be accepted by the same FSM.
- **Formal Statement**: Let ‘L’ be a regular set. Then there is a constant ‘n’ such that if ‘z’ is any word in ‘L’ such that length of ‘z’ is at least ‘n’ i.e. $|z| \geq n$, then we can write z = uvw in such a way that,
  - $|uv| \leq n$, that means, the substring near the beginning of the string is not too long.
  - $|v| \geq 1$, that means, $v \neq \in$. Since, ‘v’ is the substring that gets pumped.
  - For all $i \geq 0$, $u\,v^i\,w$ is in L. That means, the substring ‘v’ can be pumped as many times as we like and the resultant string obtained will be a member of ‘L’.
- Given a language, with the help of pumping lemma, we can determine whether it is a regular language or non-regular language.

Pumping Lemma - Example

- Prove that, the set $L = \{0^{i2} \mid i \text{ is an integer}, i \geq 1\}$ which consists of all strings of '0's whose length is a perfect square is NON-REGULAR.
- Solution: The length of each string is a perfect square.
  - Step 1: Let us assume that the language 'L' is a regular language. Let 'n' be the constant of pumping lemma.
  - Step 2: Let us choose a sufficiently large string 'z' Let $z = 0^{l2}$ for some large $l > 0$ where length of 'z' is, $|z| = l^2 \geq n$. Since, we assumed that 'L' is a regular language and is an infinite language; pumping lemma can be applied now. That means, we should be able to write 'z' as, z = uvw.
  - Step 3: As per pumping lemma every string "uv^iw", for all $i \geq 0$ is in 'L'. Also, $|v| \geq 1$, that means 'v' cannot be empty and can contain one or more symbols.
- If we consider 'v' containing any number of '0's and pumping it we will get into the situation where we will have non-square length which will not as per language definition.
- Hence, language $L = \{0^{i2} \mid i \geq 1\}$ is non-regular.

Applications of Regular Expression and Finite Automata

- Lexical analyzer
- Text Editors
- Unix *grep* command
- *Many other …*