ABHYASMITRA.IN
STUDY NOTES

THEORY OF COMPUTATIONS

Unit II — Regular Expressions and Languages

Topics Covered (09 Hours)

Introduction to Regular Expressions & Operators
Operator Precedence & Algebraic Laws for RE
Translating Informal Languages to Regular Expressions
Equivalence of Regular Expressions & Kleene's Theorem
Conversions: Regular Expressions to NFA (Thompson's Method)
Conversions: DFA to RE using Arden's Theorem Equations
Pumping Lemma for Regular Languages (Theory & Game Steps)
Closure Properties of Regular Languages (Proof Outlines)
Decision Properties (Membership, Emptiness, Finiteness)
The Myhill-Nerode Theorem (Equivalence Classes)

Compiled in a simple, easy-to-understand, book style format
for students of Computer Science and Engineering

© Copyright — abhyasmitra.in — All Rights Reserved

TABLE OF CONTENTS

2.1 Regular Expressions: Operators & Algebraic Laws

A Regular Expression (RE) is a declarative algebraic formula that describes the set of strings in a regular language. Just as arithmetic expressions use operators like addition (+) and multiplication (·) to describe sets of numbers, regular expressions use specific operators to describe sets of strings over an alphabet Σ.

Operators of Regular Expressions

There are three primary operators used to build regular expressions:

- **1. Union (or Alternation, written as + or |):** Represents choice. If R1 represents language L1 and R2 represents L2, then R1 + R2 represents the language L1 ∪ L2. For example, 'a + b' denotes the set {a, b}.
- **2. Concatenation (written as · or implicit):** Represents sequence. If R1 represents L1 and R2 represents L2, then R1 · R2 (or simply R1R2) represents the language L1L2. For example, 'ab' denotes the set {ab}.
- **3. Kleene Closure (written as *):** Represents repetition. R* represents zero or more occurrences of strings in the language represented by R. For example, 'a*' denotes the set {ε, a, aa, aaa, ...}.

Operator Precedence

To write regular expressions without excessive parentheses, we define a standard hierarchy of operator precedence. Kleene closure (*) has the highest precedence, followed by concatenation (·), and finally union (+). For example, 'a + bc*' is grouped as 'a + (b · (c*))'.

Algebraic Laws for Regular Expressions

Regular expressions obey several useful algebraic identities, which are essential for simplifying expressions and proofs:

- **Commutativity of Union:** L + M = M + L. (Concatenation is NOT commutative, i.e., LM != ML).
- **Associativity:** Union: (L + M) + N = L + (M + N). Concatenation: (LM)N = L(MN).
- **Distributivity:** Left distributive: L(M + N) = LM + LN. Right distributive: (M + N)L = ML + NL.
- **Idempotence of Union:** L + L = L.
- **Identity Elements:** ε is the identity for concatenation: Lε = εL = L. The empty set ∅ is the identity for union: L + ∅ = L. Multiplying any language by ∅ yields the empty set: L∅ = ∅L = ∅.
- **Closure Laws:** ∅* = ε. ε* = ε. L*L* = L*. (L*)* = L*.

2.2 Equivalence Conversions & Kleene's Theorem

Kleene's Theorem is a fundamental result in automata theory. It states that the class of languages accepted by Finite Automata (DFA, NFA) is identical to the class of languages described by Regular Expressions. This class is known as the Regular Languages.

RE to NFA (Thompson's Construction)

To convert a regular expression to an NFA, we build the automaton recursively from the base cases using Thompson's Construction. The rules are:

- **Base Case (ε or symbol 'a'):** A simple machine with a start state transitioning to a final state via symbol 'a' or ε.
- **Union (R + S):** Create a new start state that has ε-transitions to the start states of machine R and machine S. Create a new final state, and connect the final states of R and S to it via ε-transitions.
- **Concatenation (R · S):** Connect the final state of machine R directly to the start state of machine S via an ε-transition (or merge them).
- **Kleene Star (R*):** Create a new start state and a new final state. Add an ε-transition from the new start to the old start, and from the old final to the new final. Add a loopback ε-transition from the old final to the old start. Add a direct bypass ε-transition from the new start to the new final.

DFA to RE (Arden's Theorem)

Arden's Theorem is an algebraic tool used to solve set equations. It states that if we have an equation of the form:

R = Q + RP

Where P and Q are regular expressions and ε is not in P, then the equation has a unique solution given by:

R = QP*

To convert a DFA to a regular expression using Arden's Theorem, we set up a system of equations for each state. The variable Ri represents the language accepted starting from state qi. For each transition from qi to qj via symbol 'a', we write qi = qj · a. If qi is a final state, we add ε to its equation. We then solve the equations recursively using Arden's theorem to find the expression for the start state.

2.3 Pumping Lemma & Language Properties

Not all formal languages are regular. For example, a DFA has a finite memory (states) and cannot count arbitrary numbers. To prove that a language is NOT regular, we use the Pumping Lemma.

Pumping Lemma for Regular Languages

The Pumping Lemma states that if a language L is regular, there exists a pumping length p such that any string w ∈ L with length |w| ≥ p can be split into three substrings, w = xyz, satisfying the following three conditions:

- 1. y != ε (i.e., the pumped string y must have length |y| ≥ 1).
- 2. |xy| ≤ p (the prefix xy must be within the pumping length).
- 3. For all i ≥ 0, x(y^i)z ∈ L (the string y can be 'pumped' any number of times and the result must remain in the language).

Step-by-Step Proof of Non-Regularity

We prove a language is non-regular using proof by contradiction. Let L = {a^n b^n | n ≥ 0}. Suppose L is regular. Therefore, the pumping lemma applies with length p. We select a string w = (a^p)(b^p). Clearly, w is in L and |w| = 2p ≥ p. According to the lemma, w can be split into xyz such that |xy| ≤ p. This constraint forces the substring xy to consist entirely of 'a's. Therefore, y must consist of one or more 'a's (y = a^k, where k ≥ 1). Now we pump y by choosing i = 2. The new string is xyyz. The number of 'a's increases by k, giving a total of p + k 'a's, but the number of 'b's remains p. Since p + k != p, the string xyyz is not in L. This contradicts the pumping lemma, proving that L is NOT regular.

Closure and Decision Properties

Regular languages are closed under various operations. This means that if L1 and L2 are regular, the resulting language is also regular:

- **Closure Properties:** Regular languages are closed under Union, Intersection, Complementation, Concatenation, and Kleene Closure.
- **Decision Properties:** We can algorithmically decide key questions about regular languages: Membership (is string w in L?), Emptiness (is L = ∅?), and Finiteness/Infiniteness (does L contain a finite or infinite number of strings?).

Unit Summary

This unit explored regular expressions and the mathematical properties of regular languages.

- Regular expressions declarative formulas describing regular languages using union, concatenation, and star operators.
- Kleene's theorem establishes the equivalence of regular expressions and finite automata.
- Conversions are performed using Thompson's construction and Arden's theorem.
- The Pumping Lemma is a mathematical tool used to prove a language is not regular.
- Regular languages are closed under union, intersection, complementation, and Kleene star.

© Copyright — abhyasmitra.in — All Rights Reserved