Formal Languages & Automata Theory

Module 2 - More on Regular Languages

Regular Expression : Representation of set of strings in algebraic fashion.

Rules:

→ Any terminal symbol i.e., symbols $\in \Sigma$ including ϵ & ϕ are regular expressions

→ Union of two regular expressions is also a regular expression. $(R_1 \cup R_2)$

→ Concatenation of two regular expressions is also a regular expression. $(R_1 \cdot R_2)$

→ Closure of a regular expression is also a regular expression. R^*

→ The regular expression over Σ are precisely those obtained recursively by the application of the above rules once or several times.

Identities

1) $\phi + R = R$
2) $\phi R = R\phi = \phi$
3) $\epsilon R = R\epsilon = R$
4) $\epsilon^* = \epsilon$ & $\phi^* = \epsilon$
5) $R + R = R$
6) $R^* R^* = R^*$
7) $RR^* = R^*R = R^+$
8) $(R^*)^* = R^*$

9) $\epsilon + RR^* = \epsilon + R^*R = R^*$
10) $(PQ)^*P = P(QP)^*$
11) $(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$
12) $(P+Q)R = PR + QR$ &
$R(P+Q) = RP + RQ$

Arden's Theorem: If P and Q are two regular expressions over Σ, and if P does not contain ϵ, then the following equation in R given by $R = Q + RP$ has a unique solution i.e., $R = QP^*$.

Proof: $R = Q + RP$
$= Q + QP^*P \quad (R = QP^*)$
$= Q(\epsilon + P^*P)$
$= \underline{\underline{QP^*}} \quad (\epsilon + R^*R = R^*)$

Unique solution $\Rightarrow$
$R = Q + RP$
$= Q + [Q + RP]P$
$= Q + QP + RP^2$
$= Q + QP + [Q + RP]P^2$
$= Q + QP + QP^2 + RP^3$
$= Q + QP + QP^2 + \dots + QP^n + RP^{n+1}$
$= Q + QP + QP^2 + \dots + QP^n + (QP^*)P^{n+1}$
$= Q[\epsilon + P + P^2 + \dots + P^n + P^*P^{n+1}]$
$= QP^*$ //

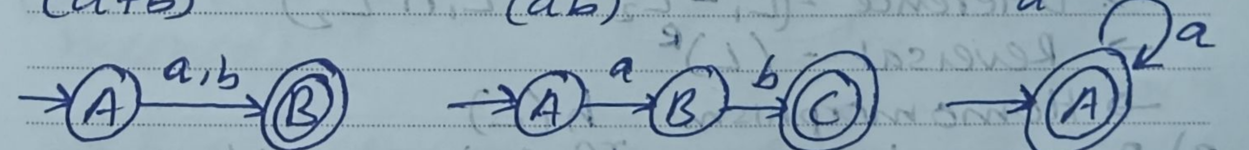
(Arden's Theorem)

Conversion of NFA / DFA to Regular expressions

1) Obtain equations for each state by considering the state and transition that lead to the chosen state.
2) Try to simplify each equation by substituting from other corresponding equation and simplify it to the Arden's form. $(R = QP^*)$
3) Repeat until there are only input symbols on the RHS.
4) Finally, Substitute this to all the final states present.
5) If there are multiple final states, take the union of both the expressions formed.

Conversion of regular expression to FA

(a+b) (ab) a*



Homomorphism $h(L)$ - Substitution function

$h(L) = \{h(w) \mid w \in L\}$

where $h: \Sigma$ to Γ^* is called homomorphism

Σ - alphabets / input symbols

Γ - ~~homo~~ symbols used by homomorphic equivalents.

Necessary conditions for regular languages

1) Should be able describe as an FA.
2) Closure Properties:
   - → Union - $(L_1 \cup L_2)$
   - → Concatenation - $(L_1 \cdot L_2)$
   - → Closure - (L^*)
   - → Complementation - $\overline{L} = \Sigma^* - L$
   - → Intersection - $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
   - → Difference - $(L_1 - L_2 = L_1 \cap \overline{L_2})$
   - → Reversal - $(L)^R$.
   - → Homomorphism - $h(L)$
3) Pumping Lemma: If A is a regular language, then A has a pumping length 'p' such that any string 's' where $|s| \geq p$ may be divided into 3 parts $s = xyz$

such that the following conditions must be true:

1) $xy^iz \in A$ for every $i \geq 0$
2) $|y| > 0$
3) $|xy| \leq p$

To prove that A is not regular:

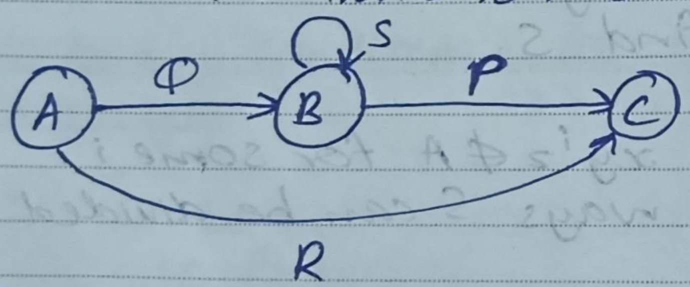
→ Assume A is regular.
→ P - pumping length.
→ $|s| \geq p$, find S.
→ $S = xyz$
→ Show that $xy^iz \notin A$ for some i.
→ Consider all ways S can be divided into xyz.
→ Show that none of these satisfy all 3 pumping conditions at same time
→ S cannot be pumped ⇒ NOT REGULAR.

Conversion of FA to RE (state elimination method)

→ Eliminate all states except initial & final

1) Choose the intermediate state to be eliminated.

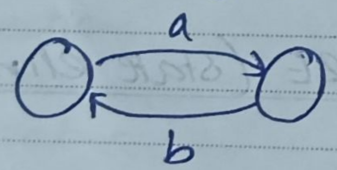
2) Bring its transitions to $\boxed{R+\Phi S^{*}P}$ form.
where R = direct transition to successor of chosen state
Φ = transition towards chosen state from predecessor state
S = transition towards chosen state itself.
P = transition towards successor.



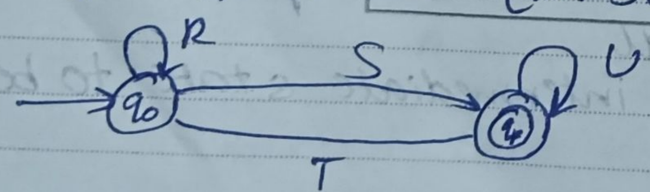
B = chosen
A = predecessor
C = successor

Substitute ϕ if there is no transition.
Repeat until there are only initial & final states.

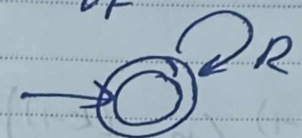
3) Consider looping paths separately, if any.



4) If $q_0 \neq q_F$, $\boxed{RE = (R+SU^{*}T)^{*}SU^{*}}$



5) If $q_0 = q_F$



$\boxed{RE = R^{*}}$

6) The desired regular expression is the union of all the expressions derived for each accepting state.

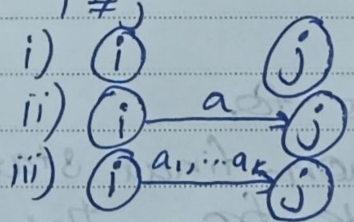
Conversion of FA to RE (Kleene's Construction / Rijik's method)

$R_{ij}^{(k)}$
R - regular expression
i - start state
j - final state
k - intermediate state

k should be $\leq$ total no. of states.

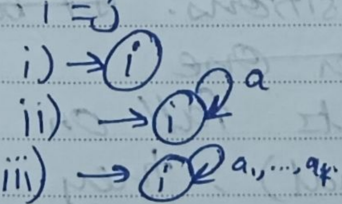
Base case. $(k=0)$

$i \neq j$



i) $R_{ij}^{(0)} = \phi$
ii) $R_{ij} = a$
iii) $R_{ij} = a_1 + a_2 + \dots + a_k$.

$i = j$



i) $R_{ij}^{(0)} = \epsilon$
ii) $R_{ij}^{(0)} = \epsilon + a$
iii) $R_{ij}^{(0)} = \epsilon + a_1 + a_2 + \dots + a_k$

$\underline{k>0}$ Induction

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

~~Identities~~ Identities (Cont.)

13) $R^* + \epsilon = R^*$
14) $(R+\epsilon)^* = R^*$
15) $(R+\epsilon)R^* = R^*(R+\epsilon) = R^*$
16) $(R+\epsilon)(R+\epsilon)^*(R+\epsilon) = R^*$
17) $R^*S + S = R^*S$
18) $\Phi + \epsilon = \epsilon$

Minimalization of DFA

Equivalence method

0 equivalence - separate sets for final and non-final state.

1st equivalence - take two states from the same set and check their transitions. ~~If both of their transitions fall on either one~~

If transitions for both inputs fall on same set (final or non-final), they can be considered equivalent.

Minimization of DFA - Table filling

Step1: Draw a table for all pairs of states

Step2: Mark all pairs where $P \in F$ & $Q \notin F$

Step3: If there's any unmarked pair (P,Q) such that $[\delta(P,x), \delta(Q,x)]$ is marked, then mark $[P,Q]$, where 'x' is an input symbol.

Step4: Repeat until no more markings can be made.

Step5: Combine all the unmarked pairs and make them a single state in the minimized DFA.