Equivalence of reg. expressions and regular languages

Theorem 1.54: A language is regular iff some regular expression describes it. (iff = if and only if)

This is yet another characterization for regular languages. We prove it in two parts:

(1) IF ($\Leftarrow$) and (2) ONLY IF ($\Rightarrow$)

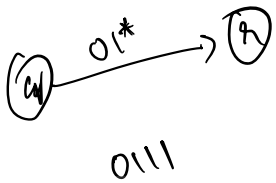
Lemma 1.55: If a regular expression describes a language, then it is regular.

In this lecture, we will prove the other direction.

Lemma 1.60: If a language is regular, then it is described by a regular expression.

Proof Outline:

Regular language $\Rightarrow$ Recognized by DFA



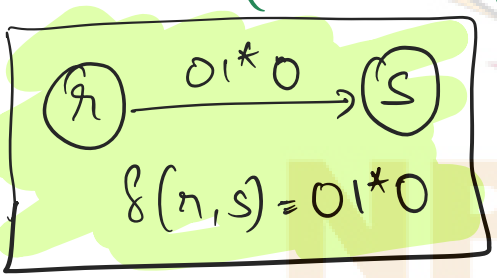
⇒ Recognized by GNFA

⇒ Described by a reg exp.

GNFA : NFA that has regular expressions on its transitions, not just symbols in Σ or ε.

Def 1.64: A generalized nondeterministic finite automaton (GNFA) is a 5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$, where

$$\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \to \mathcal{R}$$

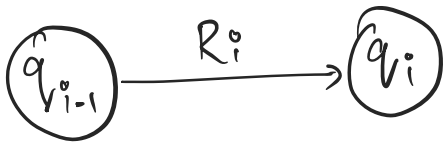


Set of all reg exp over Σ.

If $\delta(q_i, q_j) = R$, where R is some reg exp., this means that the GNFA can transition from q_i to q_j when it reads some $w \in R$.

A GNFA accepts $w \in \Sigma^*$ if w can be written as $w_1 w_2 \ldots w_k$ where $w_i \in \Sigma^*$ and there exists a sequence of states $q_0 q_1 \ldots q_k$ such that

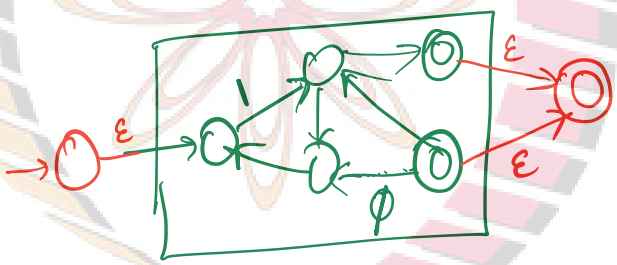
(1) $q_0 = q_{start}$



(2) $q_k = q_{accept}$

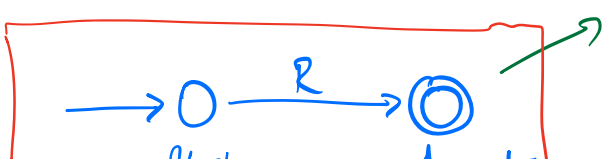
(3) For each $1 \le i \le k$, $w_i \in L(R_i)$ where

$$R_i = \delta(q_{i-1}, q_i)$$



Suppose A has a 6-state DFA. This can be converted to an 8-state GNFA. Then we convert this into a 7-state GNFA, 6-state GNFA, and so on till we reach a 2-state GNFA.

2-state GNFA

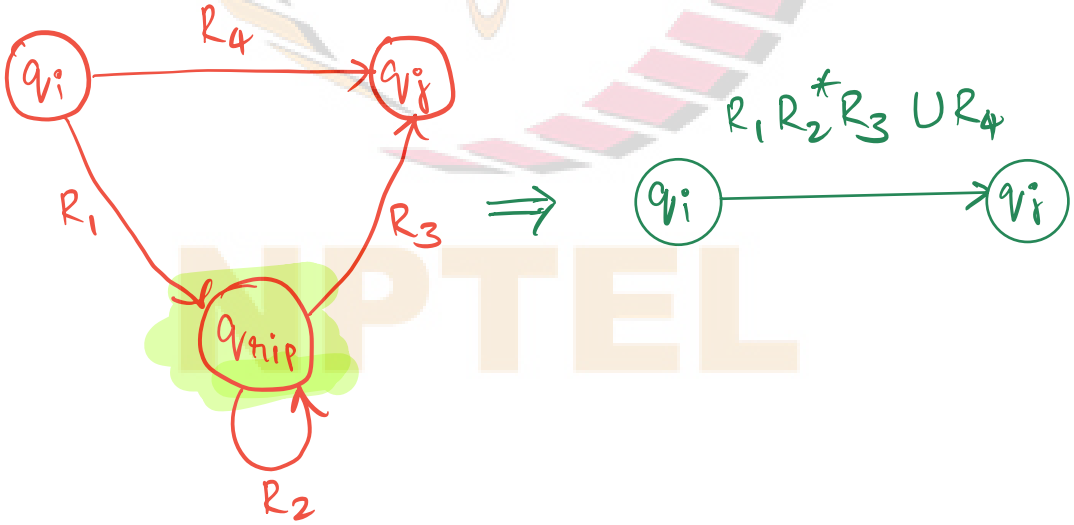


R is the equiv. reg. exp.

Then we get $L(R) = A$.

Main ideas of the proof.

The key idea of the proof is in reducing the number of states of the GNFA by one.



G is the GNFA

Convert (G): Returns R or recursively calls itself

1. k = no. of states of G

2. If $k = 2$, then return R.

3. If $k > 2$, then select a state q_{rip} such that $q_{rip} \in Q \setminus \{q_{start}, q_{accept}\}$.

   For all pairs q_i, q_j which is not q_{rip}, do the above transformation.

   $G' = (Q', \Sigma, \delta', q_{start}, q_{accept})$

   $Q' = Q \setminus \{q_{rip}\} = Q - \{q_{rip}\}$

   $\delta'(q_i, q_j) = R_1 R_2^* R_3 \cup R_4$

4. Call Convert (G').

Claim 1.65: For any GNFA G, convert (G) is equivalent to G.

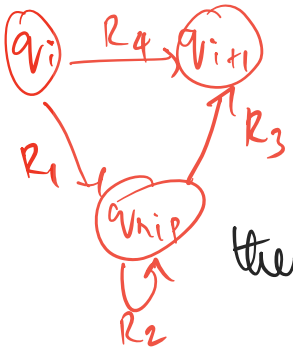
Proof: If $k = 2$, this is true by the definition of

GNFA. If $k > 2$, we will show that G' is equivalent to G.

Suppose G accepts w. Let G go through the states $q_{start}\, q_1, q_2 \cdots q_{accept}$.

→ If q_{rip} is not in the above sequence, then G' accepts w because all the transitions remain in G'.

→ If q_{rip} is present between q_i and q_{i+1},



$q_i\ q_{rip}\ q_{rip}\ q_{rip}\ q_{i+1}$

then $R_1 R_2^* R_3 \cup R_4$ describes this as well.

This shows that $G' = Connect(G)$ is equivalent to G. By using induction, we can reason that G' is equivalent to the final 2-state GNFA.

This shows that G is also equivalent to the

2-state GNFA and hence to K as well.

This completes the proof that regular expressions are another way to characterize regular languages.