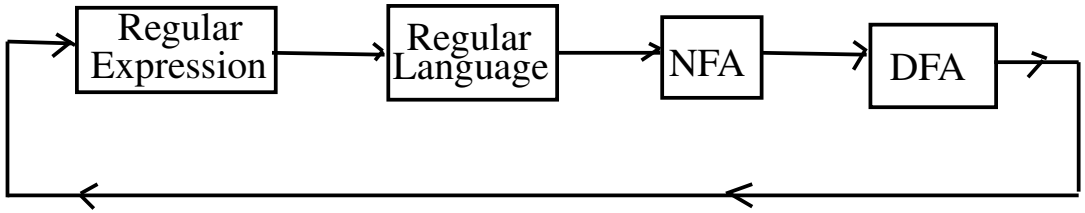
Regular Languages and their Properties

Prof. (Dr.) K.R. Chowdhary
Email: kr.chowdhary@iitj.ac.in

Formerly at department of Computer Science and Engineering
MBM Engineering College, Jodhpur

Tuesday 6th December, 2016

Introduction to Regular languages



- There is a cycle: for regular expression there is corresponding regular language, for reg. language there is a corresponding NFA, for NFA there is a corresponding DFA, and for that there is a corresponding regex. It indicates that all these have same capabilities.
- We know that regular expressions are closed on $\cup, \circ$, and $*$. Do these properties also apply to Reg, languages?

Regular languages are closed on $\cup, \circ$, and $*$

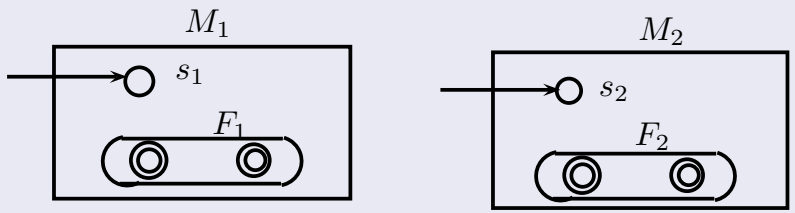
The approach for this is: If after performing these operations on reg. languages, if they are still accepted by FA, then they (reg. languages) are closed. This is because, the FA recognize regular languages. **Thus, the proof reduces only to: find corresponding FA** for these closure properties.

Theorem

Regular languages are closed on Union.

Proof.

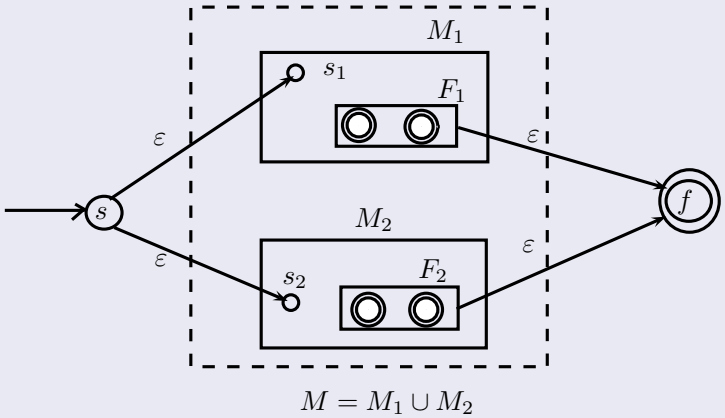
Assume that $M_1 = (Q_1, \Sigma_1, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, s_2, F_2)$ are two FA recognizing the regular languages $L_1 = L(M_1)$ and $L_2 = L(M_2)$. Let $L(M) = L(M_1) \cup L(M_2)$ is the language recognized by a union automaton, such that, $L(M) = \{w | w = x \cup y, x \in L_1, y \in L_2\}$. That is, M accepts w iff M_1 accepts x or M_2 accepts y.



Regular languages Theorem

Proof.

Closed on Union: Let the automata M which accepts $L_1 \cup L_2$ is as shown below.



If $w = x$, the *FA* M non-deterministically decides a ε-transition from new state s to s_1, and then there are transitions in M_1, and when $f_1 \in F_1$ is encountered there is a $\varepsilon-$transition from f_1 to f. Similar phenomena occurs for M_2 when $w = y$.
The resultant FA $M = (Q, \Sigma, \delta, s, F)$ is given as: $Q = Q_1 \cup Q_2 \cup \{s, f\}$, $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{\varepsilon\}$, start state=$s$, final state=$f$, and δ is given in next slide.

Proof.

Closed on Union contd.: The δ

$$\delta(q,a) = \delta_1(q,a), \text{ if } q \in Q_1 \text{ and } a \in \Sigma_1$$
$$\delta(q,a) = \delta_2(q,a), \text{ if } q \in Q_2 \text{ and } a \in \Sigma_2$$
$$\delta(q,a) = \{s_1, s_2\}, \text{ if } a = \varepsilon \text{ and } q = s$$
$$\delta(q,a) = \delta(q,a) = f, \text{ if } q \in F_1 \cup F_2 \text{ and } a \in \varepsilon$$

hence

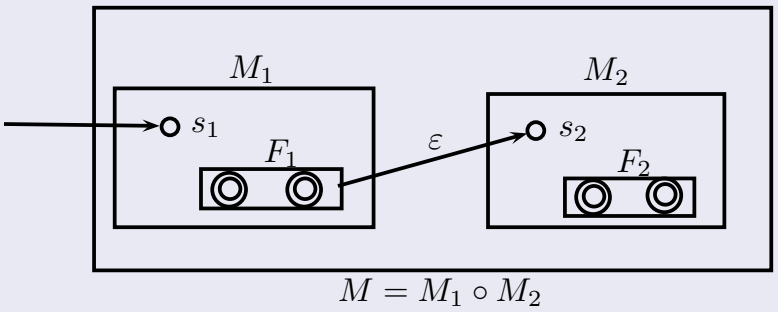
$$L = L(M) = \{x \cup y \mid x \in \Sigma_1^*, \text{ and } y \in \Sigma_2^*\}.$$

Regular languages theorem

Theorem

Regular languages are closed on the operation of concatenation.

Proof.



Let $L = L(M_1) \circ L(M_2) = \{w = x \circ y \mid x \in L(M_1), y \in L(M_2)\}$. Here, M_1 recognizes x and reaches to its final state, then there is a ε-transition from $f_1 \in F_1$ to s_2, then M_2 recognizes y. M is defined in terms of M_1 and M_2 as: $Q = Q_1 \cup Q_2$, $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{\varepsilon\}$, $s = s_1$, $f = f_2$ and δ is:

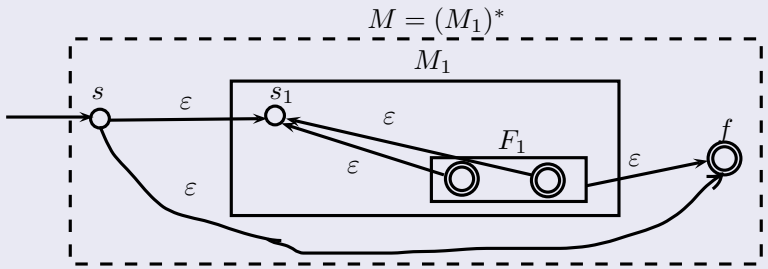
$$\delta(q,a) = \delta_1(q,a), q \in Q_1, a \in \Sigma_1$$
$$\delta(q,a) = \{s_2\}, q \in F_1, a = \varepsilon$$
$$\delta(q,a) = \delta_2(q,a), q \in Q_2, a \in \Sigma_2.$$

Regular languages theorem

Theorem

Regular languages are closed on the operation of kleene-star.

Proof.



This is extension of concatenation, i.e., the input $x \in L(M_1)$ is pumped zero or more number of times. $L = L(M) = (L(M_1)^*) = L_1^* = \{w = x^* \mid x \in L_1\}$. The formal definition of $M = (Q, \Sigma, \delta, s, F)$ is:

$Q = Q_1 \cup \{s, f\}, \Sigma = \Sigma_1 \cup \{\varepsilon\}$, start state=$s$, final state=$f$, and δ is:

$\delta(q, a) = \{s_1, f\}$, if $q = s, a = \varepsilon$,

$\delta(q, a) = \delta_1(q, a)$, if $q \in Q_1, a \in \Sigma_1$,

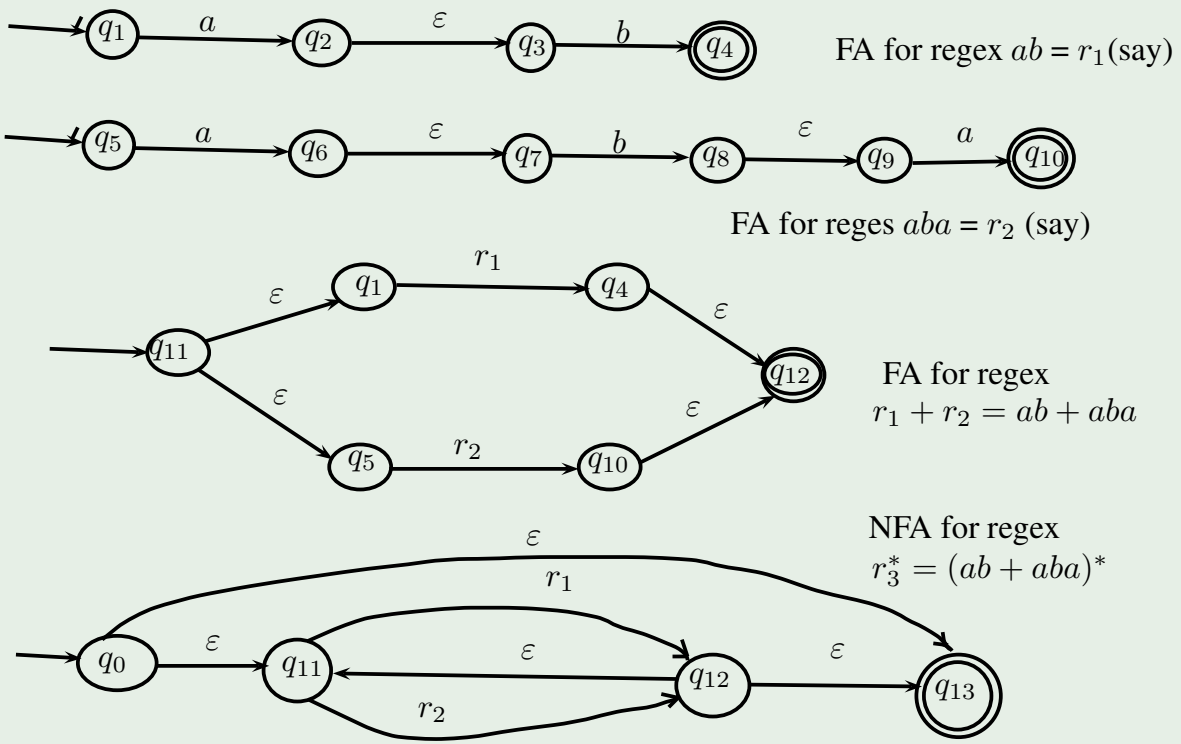
$\delta(q, a) = \{s_1 \cup f\}$, if $q \in Q_1, a = \varepsilon$

This proves the theorem.

Construction of FA for a given Regex

Example

Given the regex $r = (ab + aba)^*$, we can find the FA as follows: Systematically apply the theorem discussed above to construct the FA using elementary steps. 1. Construct the NFA for regex $r_1 = ab$, 2. Construct NFA for $r_2 = aba$, 3. Construct NFA for $r_3 = r_1 + r_2 = ab + aba$, 4. Finally, construct NFA for $r_3^* = (r_1 + r_2)^* = (ab + aba)^*$



FA to Regex: State removal method

Lemma

A language is regular then it can be described by a regex.

Proof.

1. Because L is regular, it can be represented by a *DFA*.
2. Convert the *DFA* into equivalent regex.
3. First convert DFA into GNFA (Generalized NFA) then convert GNFA into regex. This proves the lemma.

□

Definition

(GNFA.) A GNFA G reads blocks of symbols rather than single symbol at a time, has single start state and single final state. If GNFA is $G = (Q, \Sigma, \rho, q_0, q_f)$, then $\rho : Q - \{q_0\} \times Q - \{q_f\} \to R_\Sigma$, R_Σ is set of all regular expressions over alphabet Σ.

FA to Regex: State removal method

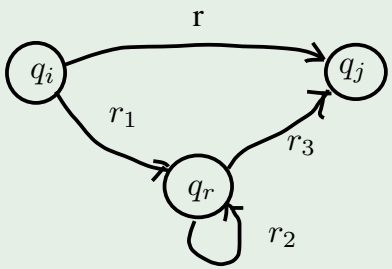
DFA-to-GNFA:

1. add new start state (q_{start}) with $\varepsilon-$transition to original start state q_0
2. add new end state with $\varepsilon-$transition from each of the original final state.
3. If original transition arrow has multiple labels, we replace these with a new arrow whose label is regular expression formed by the union of the labels.
4. we iteratively remove one state in GNFA, such that new GNFA obtained will recognize the same language.
5. When number of states in GNFA are two, we have obtained the regex for the given DFA.

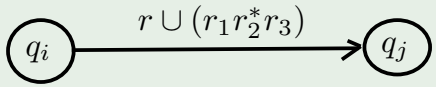
FA to Regex: State removal method

Example

The diagram below shows state removal and gives regex for a FA.



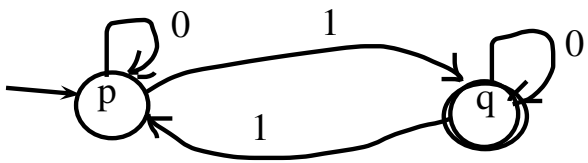
Original FA



FA after removal of q

FA to Regex Brozozowski's method

This method generates regex by generating sequence of symbols (or equivalently, scanning the sequence of symbols). Suppose we want to find the regular expression for the FA given below.



Starting from each state we we write a equation showing the generating of sequence of state and edge labels. Finally, after doing substitutions, we keep the equation of start state, without any variables(i.e., other state symbols).

$$p = 0p + 1q$$

$q = 0q + (1p + \varepsilon)$; ε symbols indicates that when input is accepted, no transition is required. So,

$q = 0^*(1p + \varepsilon)$; by Arden's theorem: $X = X1 + Y = X^*Y$, so

$$q = 0^*1p + 0^*(\varepsilon)$$

$$= 0^*1p + 0^*$$

$$p = 0p + 1q$$

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

$$= 0p + 10^*1p + 10^*$$

$= (0 + 10^*1)^*(10^*)$; by Arden's rule.

Hence, the required regex is $(0 + 10^*1)^*(10^*)$

Kleene's theorem

Theorem

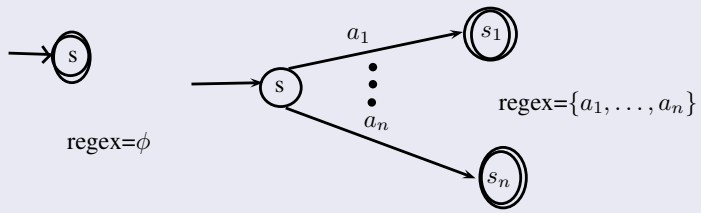
A language is regular iff it is accepted by FA.

Proof.

The proof has two parts: 1. If the language is regular then it is accepted by FA, and 2. If a language is accepted by FA then it is regular.

Part 1: By definition of regularity, following are regular: $\phi, \{a_1, a_2, \ldots, a_n\}, \{a_i\} \cup \{a_j\}, \{a_i\} \circ \{a_j\}, a_i^*$. The diagrams shown below are of FA recognizing the first two.

We have already proved that FA exists for the following regular languages: $\{a_i\} \cup \{a_j\}, \{a_i\} \circ \{a_j\}, a_i^*$.



Part 2: Using the method of state reduction (converting to GNFA) and Brozozowki methods we have proved that there exists regex for every FA. Hence, the complete proof.