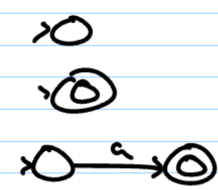
Every language generated by a regular expression is accepted by some NFA and vice versa.

Given reg exp, construct corresponding NFA:

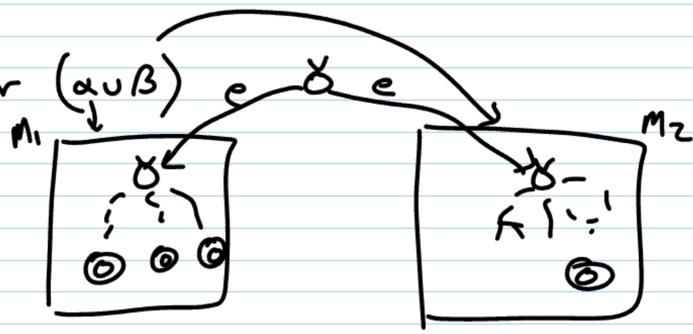
1) NFA for $\emptyset$: $\emptyset^* = \{e\}$

e :

a :



2) NFA for $(\alpha \cup \beta)$



Formally, find $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ that accepts $\mathcal{L}(\alpha)$

$M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ that accepts $\mathcal{L}(\beta)$

assume WLOG $K_1 \cap K_2 = \emptyset$

then $M = (K_1 \cup K_2 \cup \{q_0\}, \Sigma, \Delta_1 \cup \Delta_2 \cup \{(q_0, e, s_1), (q_0, e, s_2)\}, q_0, F_1 \cup F_2\}$

accepts $\mathcal{L}(\alpha \cup \beta))$

If $(\alpha \cup \beta)$ generates w then either α generates or β does.

If α then M_1 accepts w by some path

$$(s_1, w) \vdash (q_1, w_2 \cdots w_k) \vdash \cdots \vdash (f, e)$$

transitions were included in M

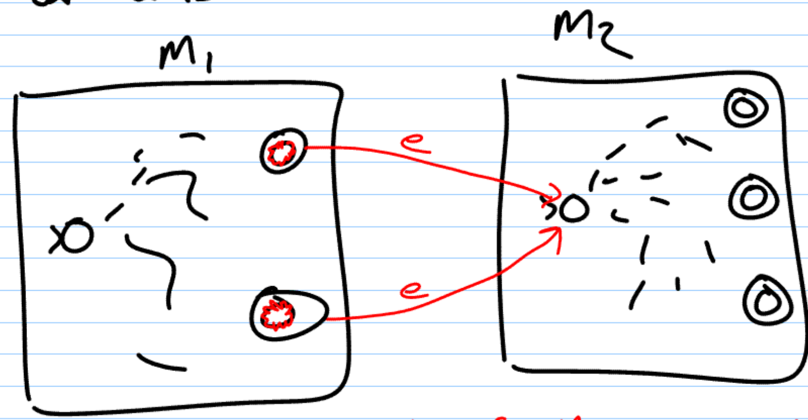
and so M accepts w by path

$$(q_0, w) \vdash (s_1, w) \vdash (q_1, w_2 \ldots w_k) \vdash \cdots \vdash (f, e)$$

If β generated w then ---- ...

If M accepts w then - - - - - (work backwards from)

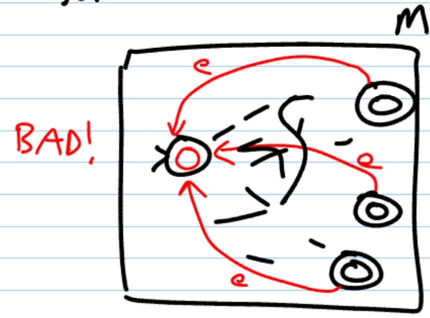
3) NFA for $\alpha \circ \beta$

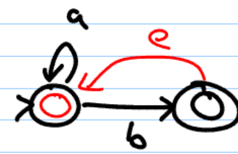


add e-transitions from M_1's accepts to M_2's start

make M_1's accepting states non-accepting

4) NFA for α^*

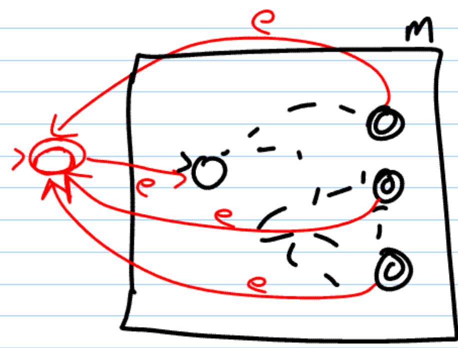




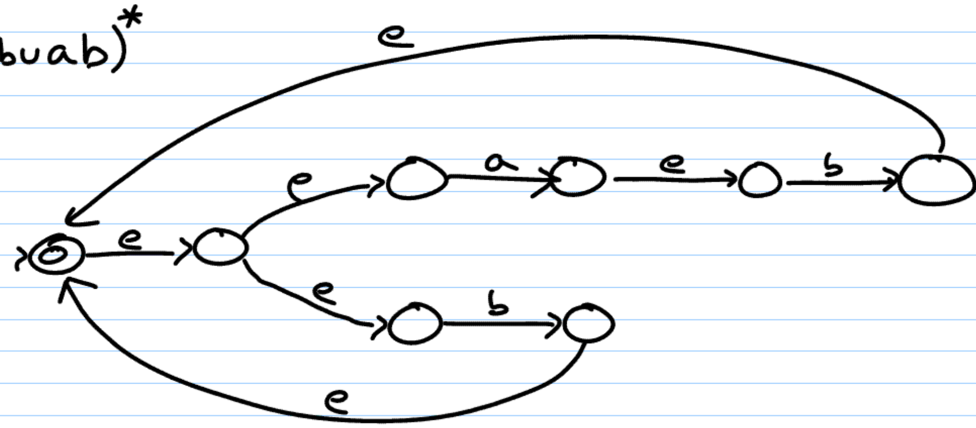
a^*b

$(a \cup b)^* \neq (a^*b)^*$

aaaa



$(b \cup ab)^*$



Given NFA M, construct regular expression that generates $\mathcal{L}(M)$:

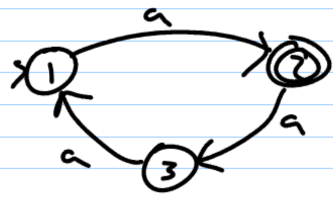
Number states $1 \ldots n$

$R(i, j, k)$ = regular expression for set of strings that drives M from q_i to q_j w/o using in the middle states q_{k+1} or above

if $s = q_1$ and final state is q_n

then $R(1, n, n)$ is regular exp for entire machine

$R(i, j, 0) =$



$R(1,2,0) = a$

$R(2,3,0) = a$

$R(1,3,0) = \emptyset$