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

Given reg exp, construct corresponding NFA:

1) NFA for $\emptyset$:

$\emptyset^* = \{e\}$

- $e$ : $\emptyset$
- $a$ : $\emptyset \xrightarrow{a} \emptyset$

2) NFA for $(a \cup b)$:

- $M_1$ to $M_2$ with $e$ transitions between them.
Formally, find $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ that accepts $L(\alpha)$

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

Assume w.l.o.g. $K_1 \cap K_2 = \emptyset$

Then $M = (K_1 \cup K_2 \cup \{g_0\}, \Sigma, \Delta \cup \Delta_2 \cup \{(g_0, e, s_1), (g_0, e, s_2)\}, g_0, F_1 \cup F_2)$

accepts $L(\alpha \cup \beta)$

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

If $\alpha$ then $M_1$ accepts $w$ by some path

$$(s_1, w) \xrightarrow{(g_1, w_1, \ldots, w_k)} \ldots \xrightarrow{(f, e)}$$

and so $M$ accepts $w$ by path

$$(g_0, w) \xrightarrow{(g, w_1, \ldots, w_k)} \ldots \xrightarrow{(f, e)}$$

If $\beta$ generated $w$ then --- ...
If M accepts w then ..... (work backwards from)

3) NFA for $a \cdot B$

add e-transitions from M1's accepts to M2's start
make M1's accepting states non-accepting
4) NFA for $L^*$

$\text{BAD!}$

$a^*b$

$(a+b)^* \neq (a^*b)^*$
Given NFA $M$, construct regular expression that generates
Number states $1, \ldots, n$

$L(M) = \sum_{i, j, k} R(i, j, k)$

Regular expression for set of strings
that drives $M$ from $q_i$ to $q_j$ w/o
in the middle
using states $q_{k+1}$ or above

if $s = q_i$ and final state is $q_n$
Then $R(1, n, 0)$ is regular exp for entire machine

$R(i, j, 0) =$

- $R(1, 2, 0) = a$
- $R(2, 3, 0) = a$
- $R(1, 3, 0) = \emptyset$