Relation between RE and FA

Sagar Giri && Anju Shahi

Conversion from DFA to RE

- Transitive Closure Method
  - Rather theoretical approach.
  - Sketch of the method:
    - Let Q = {q_1 , q_2 , . . . , q_m } be the set of all automatons states.
    - Suppose that regular expression R_{ij} represents the set of all strings that transition the automaton from q_i to q_j .
    - Wanted regular expression will be the union of all R_{sf} , where q_s is the starting state and q_f is one the final states.
  - • The main problem is how to construct R_{ij} for all states q_i q_j.

How to construct Rij ?

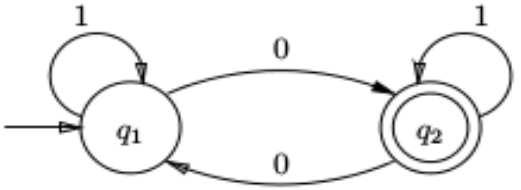
R_{ij}^k is recursively defined as:

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

Assuming we have initialized R_{ij}^0 to be:

$$R_{ij}^0 = \begin{cases} r & \text{if } i \neq j \text{ and r transitions NFA from } q_i \text{ to } q_j \\ r + \varepsilon & \text{if } i = j \text{ and r transitions NFA from } q_i \text{ to } q_j \\ \emptyset & \text{otherwise} \end{cases}$$

Example



$$
\begin{aligned}
r_{11}^0 &= 1+\epsilon \\
r_{12}^0 &= 0 \\
r_{12}^1 &= r_{12}^0 + r_{11}^0 (r_{11}^0)^* r_{12}^0 \\
&= 0 + (1+\epsilon)^+ 0
\end{aligned}
\qquad
\begin{aligned}
r_{22}^0 &= 1+\epsilon \\
r_{21}^0 &= 0 \\
r_{22}^1 &= r_{22}^0 + r_{21}^0 (r_{11}^0)^* r_{12}^0 \\
&= (1+\epsilon) + 0(1+\epsilon)^* 0
\end{aligned}
$$

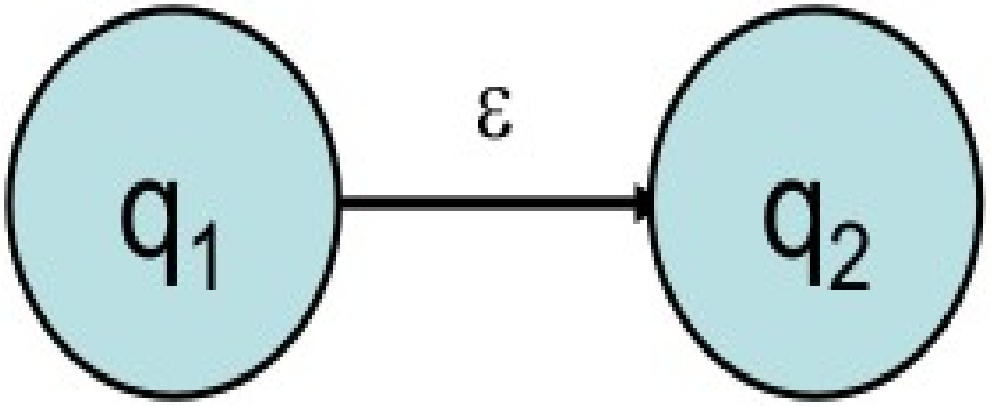
$$
\begin{aligned}
r_{12}^2 &= r_{12}^1 + r_{12}^1 (r_{22}^1)^* r_{22}^1 \\
&= (0 + (1+\epsilon)^+ 0) + (0 + (1+\epsilon)^+ 0)((1+\epsilon) + 0(1+\epsilon)^* 0)^+ \\
&= (0 + (1+\epsilon)^+ 0)(1 + \epsilon + 0(1+\epsilon)^* 0)^* \\
&= (1+\epsilon)^* 0 (1 + \epsilon + 01^* 0)^* \\
&= 1^* 0 (1 + 01^* 0)^* \\
L(M) &= L(r_{12}^2)
\end{aligned}
$$

Next Part:

Conversion from RE to NFA

First of all.... ε-NFA

- transitions made “for free”, without “consuming” any input symbols.



Finite Automata and Regular Expression

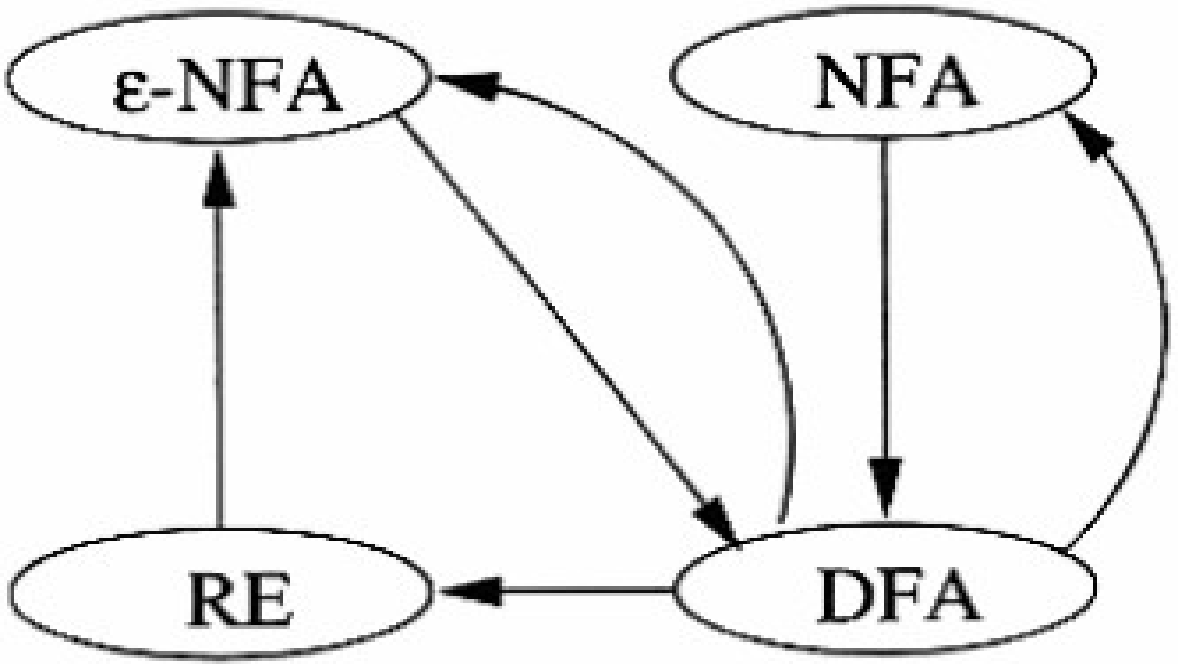
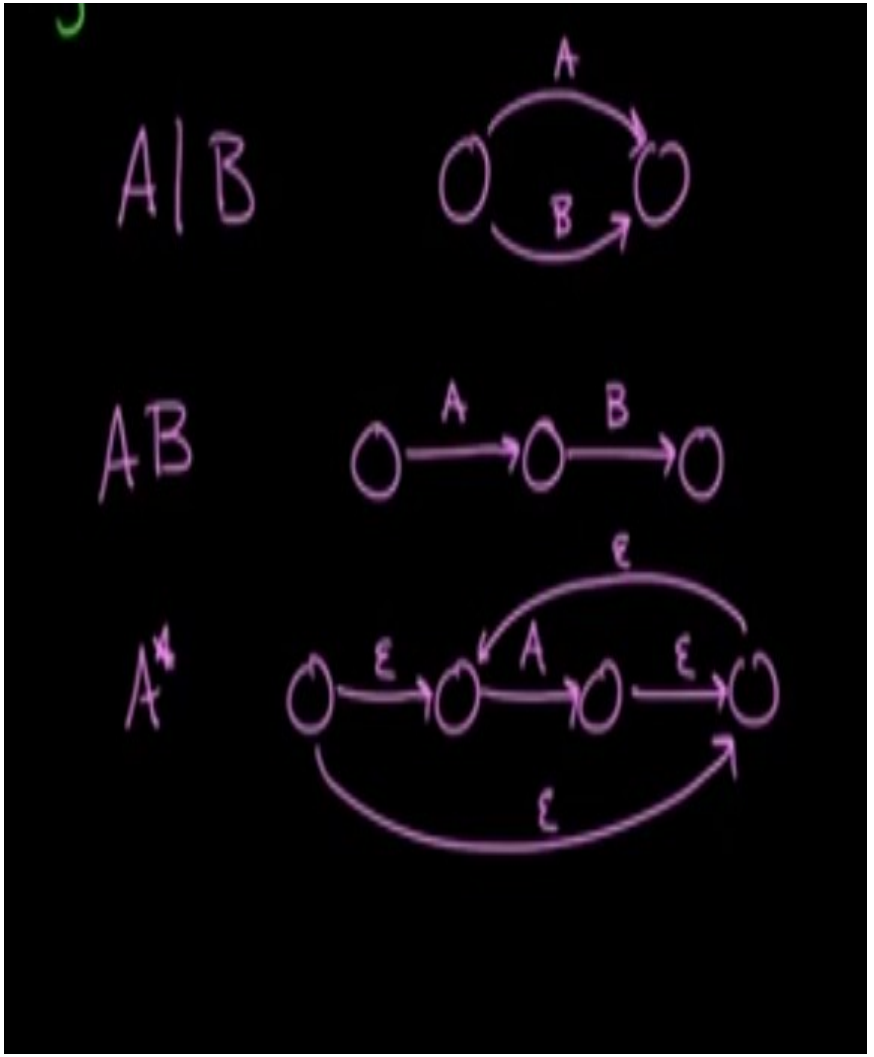
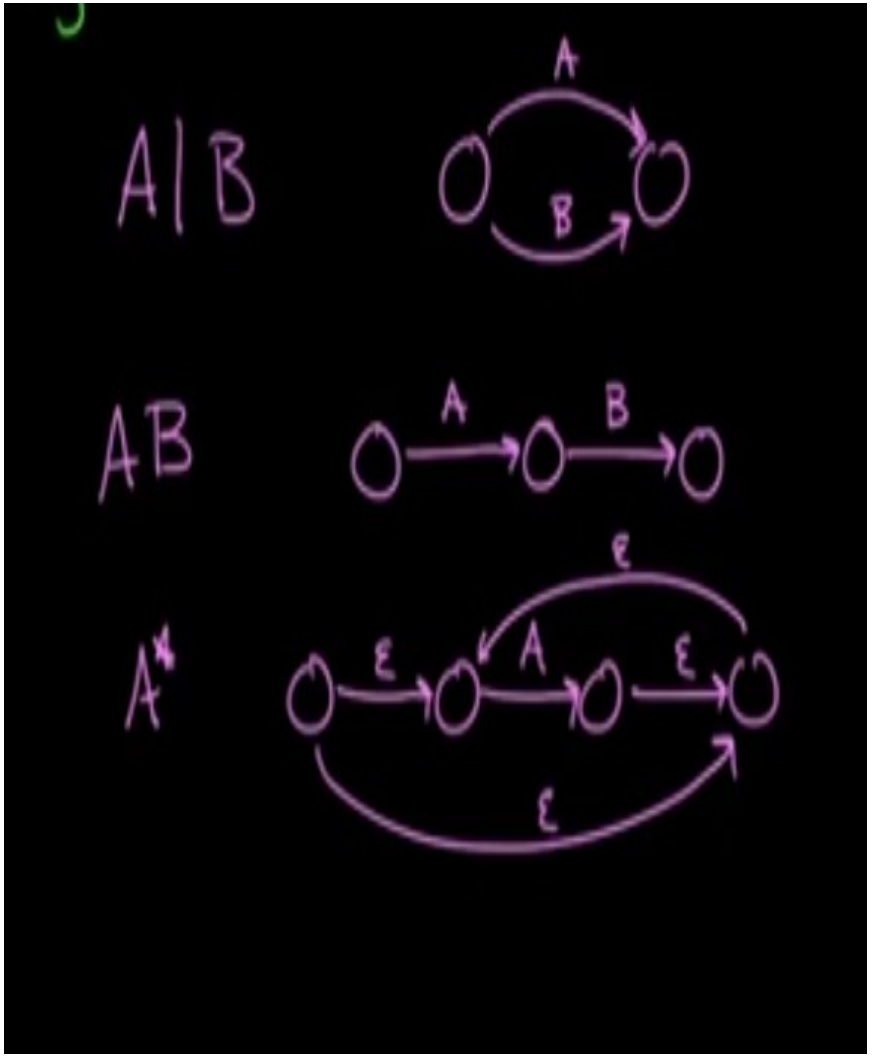


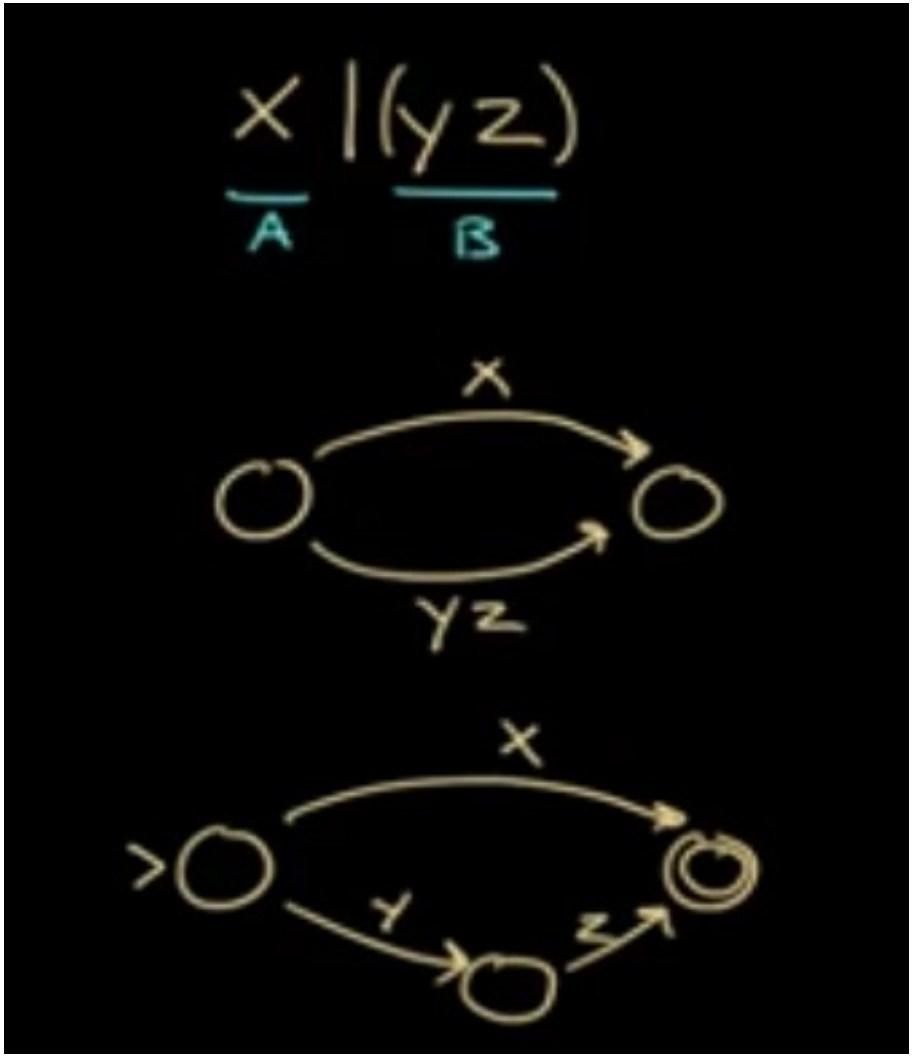
Figure 3.1: Plan for showing the equivalence of four different notations for regular languages

Conversion from RE to NFA

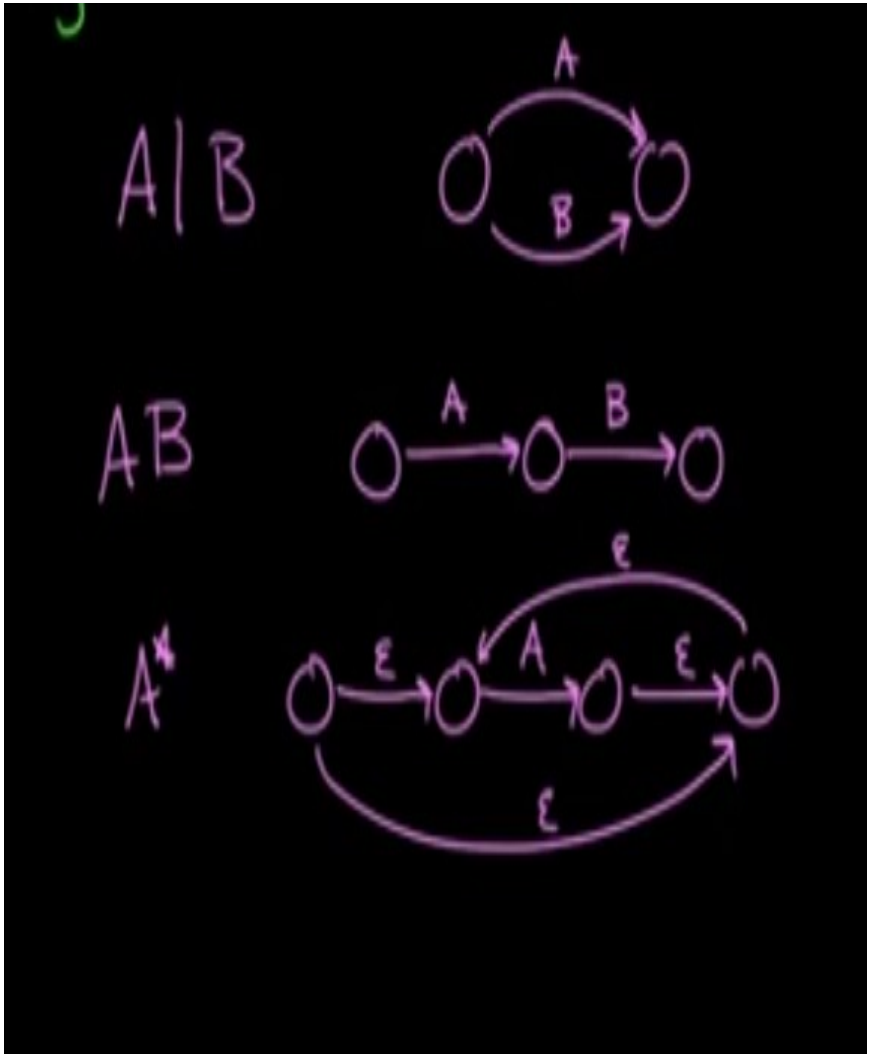


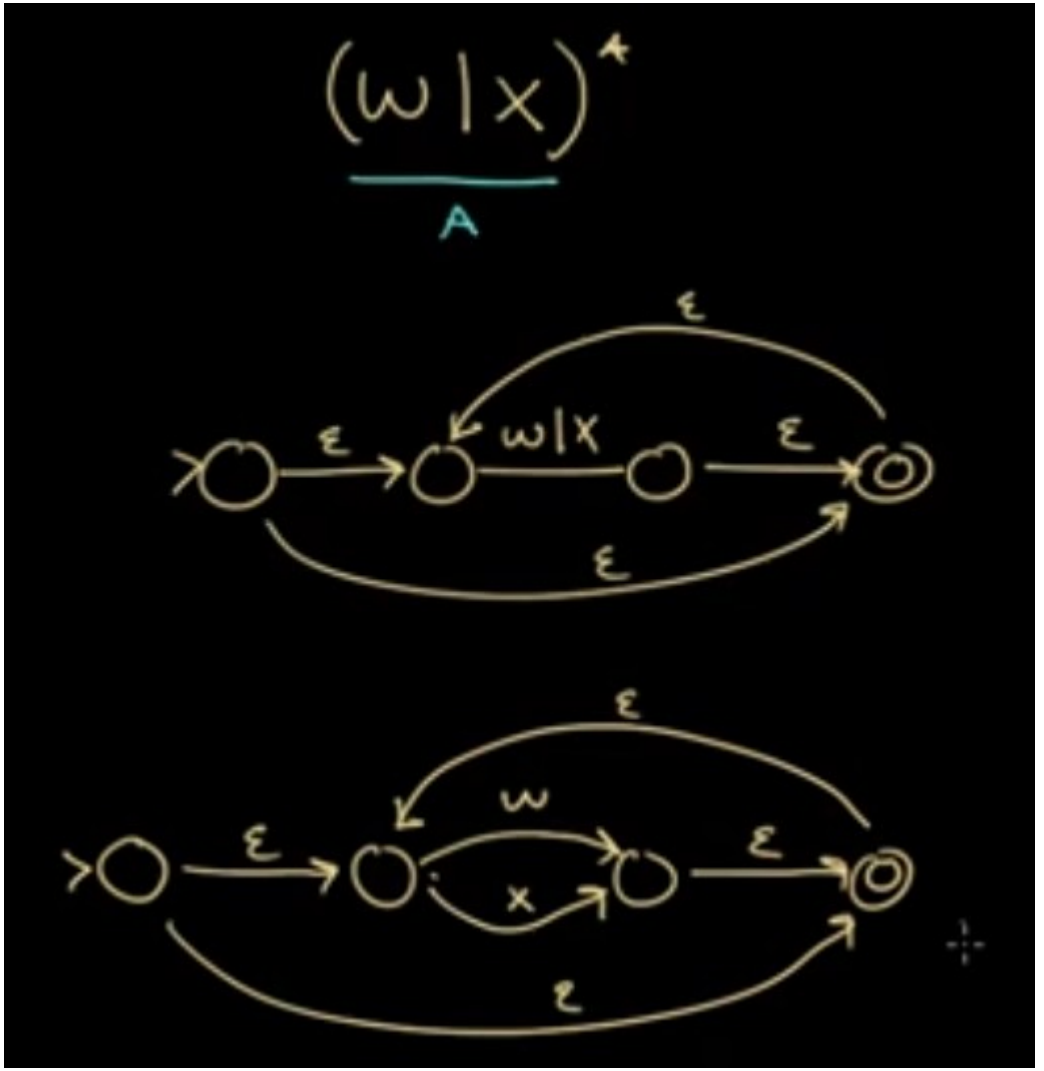
Conversion from RE to NFA



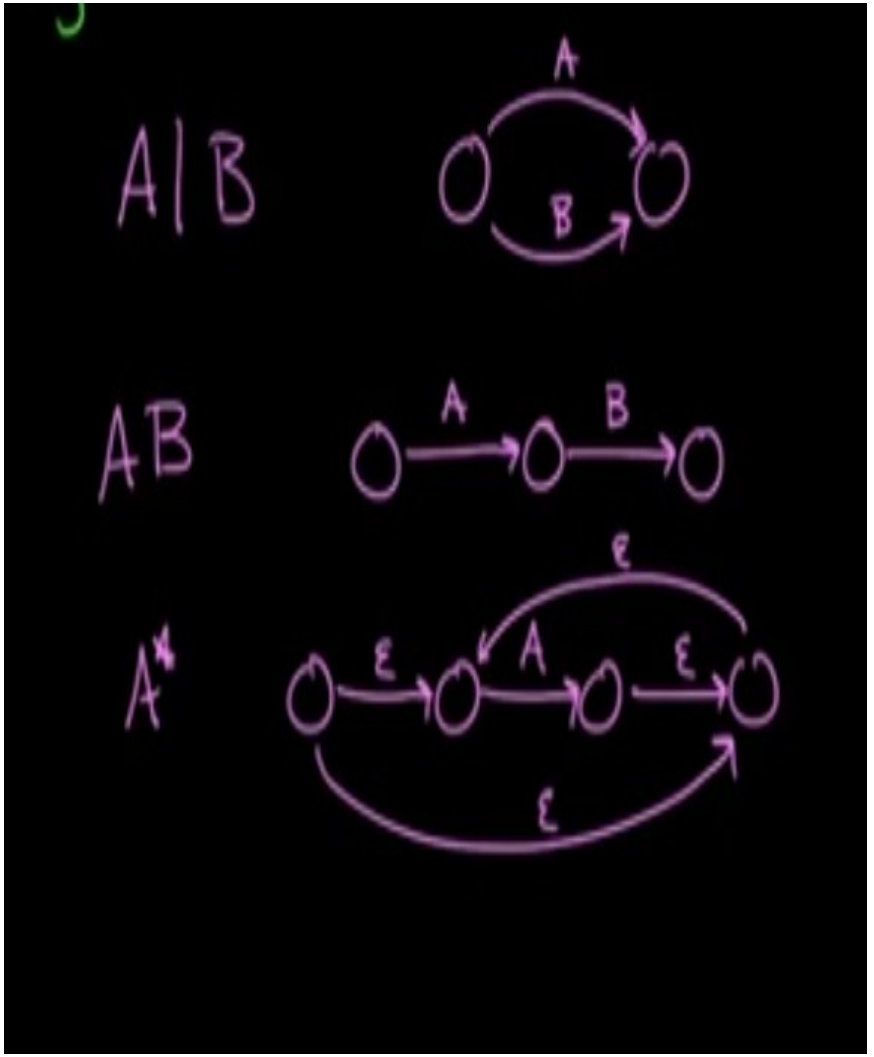


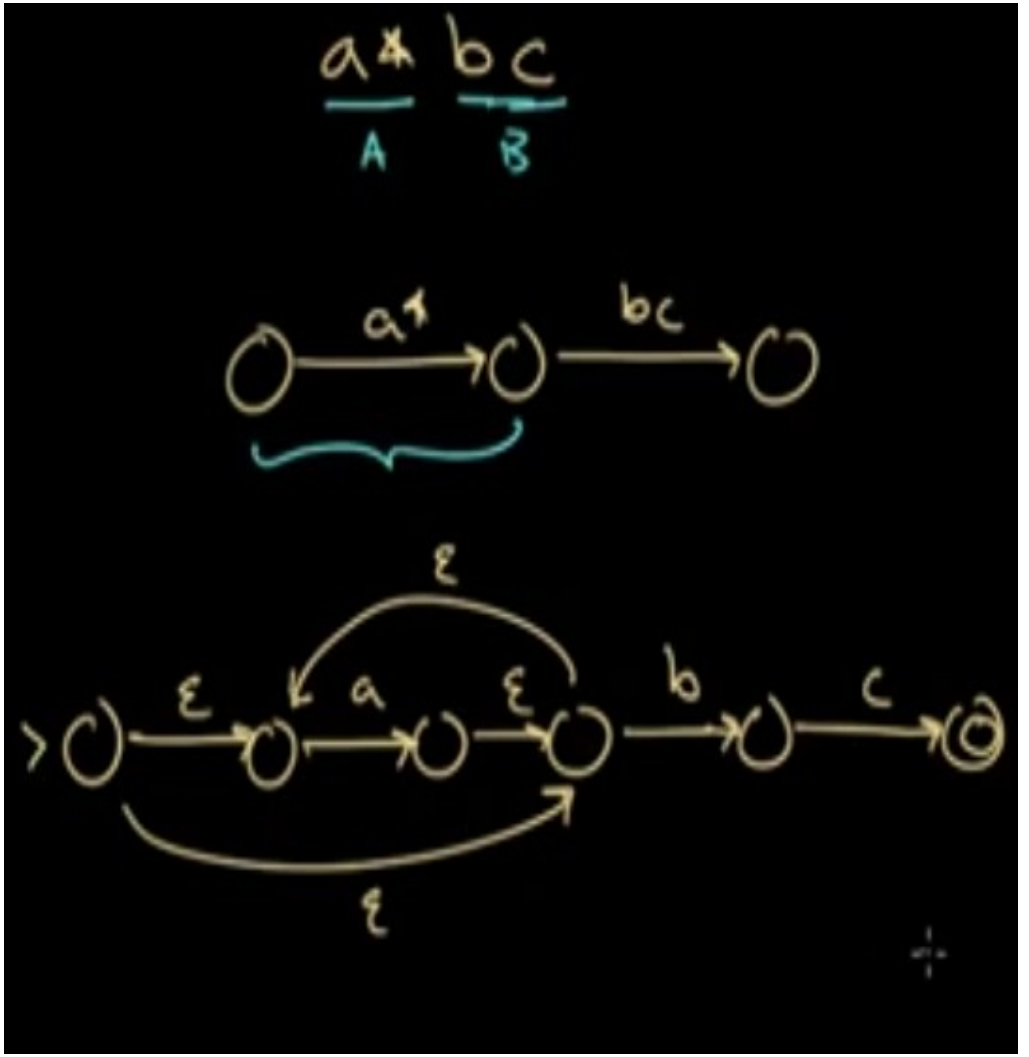
Conversion from RE to NFA



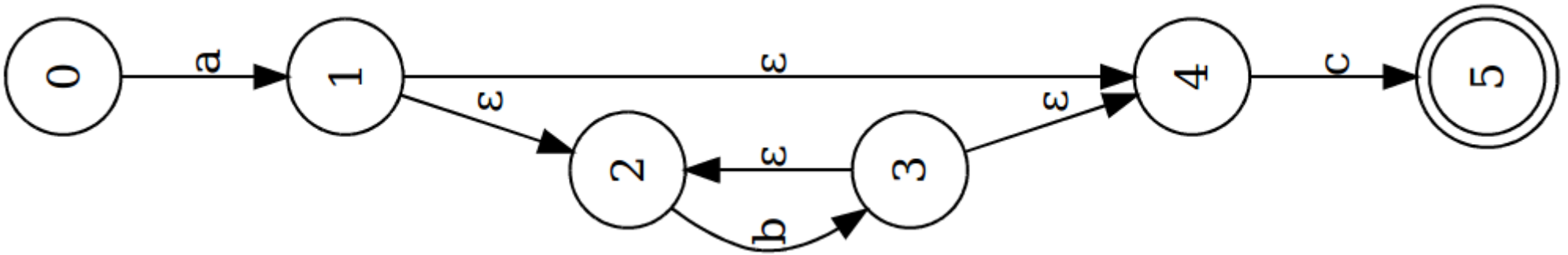


Conversion from RE to NFA





For ab*c



NFA

http://hackingoff.com/compilers/regular-expression-to-nfa-dfa

END