DFA

$M = (\Sigma, Q, \delta, s, A)$

$\delta : \Sigma \times Q \to Q$

$\delta(q, a)$ $\delta^*(q, \text{string})$

NFA

$M = (\Sigma, Q, \delta, s, A)$

$\delta : \Sigma \times Q \to 2^Q$

$$\delta^*(q, w) = \bigcup_{r \in \delta(q,a)} \delta(r, x)$$

(whenever $w = rx$)

NFA's have ε-transitions

Regex

$+$ = union operator

$\bullet$ = concat operator

S^* = Kleene Star + ε

S^+ = Kleene Star − ε

Product Construction

$Q = Q_1 \times Q_2$

$A = \{(q_1, q_2) \ldots \mid q_1 \in Q_1, q_2 \in Q_2\}$

$\delta : Q \times Q \times a \to Q \times Q$

Fooling Sets

- have set of strings, such that $a \notin L$ $\forall a \in$ fooling set, but ~~one~~ every element has a suffix that makes it ~~the~~ distinguishable from others: $a + s \in L$ but $b + s \notin L$ $\forall b \in F$, $b \neq a$
- Infinite fooling set → not regular

NFA → DFA "Subset construction"

Make Table of columns:

- DFA state
- ε-reach of corresponding
- Whether state is accepting
- output of transition function $\forall a \in \Sigma$

$q \mid \varepsilon\text{-reach} \mid q \in A? \mid \delta'(q, a) \mid \delta(q, b) \ldots$

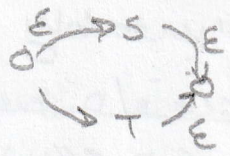
DFA/NFA → regex "State Removal Method"

- I starting state is accepting
- Turn all "A" into non-accepting states, connect them via ε to an explicit accepting state
- remove all non-accepting states via state removal until you have: $\to \circ \to \circledcirc$

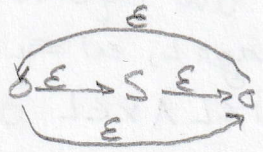
Regex → NFA "Thompson's Algorithm"

$R = ST \to$ $S \xrightarrow{\varepsilon} T$

$R = S + T \to$



$R = S^* \to$



Context free grammar

$G = (\Sigma, T, \rightsquigarrow, S)$

S = starting non-terminal

Σ = terminals

T = non terminals

$\rightsquigarrow$ = production rules