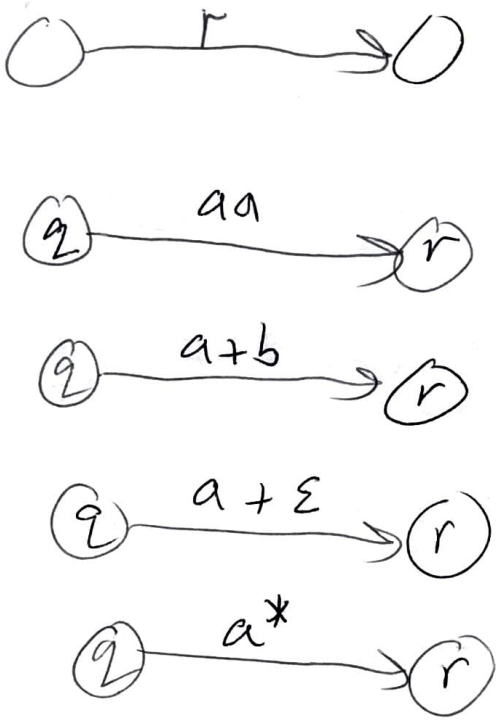
CSCE 355
2/9/2022

ε-NFA $\Longrightarrow$ regex

Idea: Given an ε-NFA
build a sequence of generalized NFAs (GNFAs)



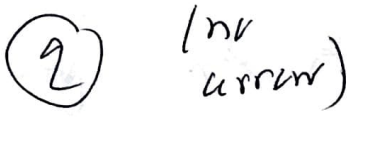
Def: For any alphabet Σ, we let REG_Σ be the set of all regexes over Σ.

Def: A generalized NFA (GNFA) is a 5-tuple $\langle Q, \Sigma, \delta, q_0, F \rangle$ where Q, Σ, q_0, F are as with an ε-NFA (or NFA or DFA) and

$$\delta : Q \times Q \longrightarrow REG_\Sigma .$$

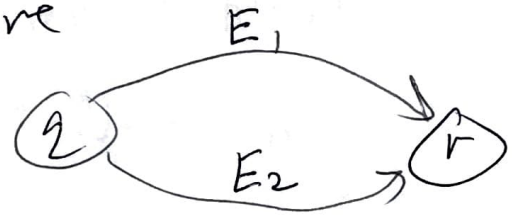
(2) For the transition diagram,

means $E = \delta(q, r)$

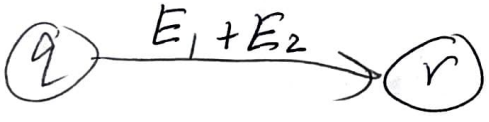


means $\delta(q, r) = \emptyset$

Can't have



instead



⇓ edge consolidation

Def: Let $G = \langle Q, \Sigma, \delta, q_0, F \rangle$ be a GNFA and let $w \in \Sigma^*$ be a string over Σ. A (complete) computation path of G on input w is a sequence of states $s_0, s_1, \ldots, s_k$ $(k \geq 0)$ such that there exist strings $w_1, w_2, \ldots, w_k \in \Sigma^*$ where:

1) $w = w_1 \cdots w_k$
2) $s_0 = q_0$
3) $\forall i,\ 1 \leq i \leq k,\ w_i \in L(\delta(s_{i-1}, s_i))$

We say the path ends in state s_k

(3)

Def: Let G and w be as in the prev. def.

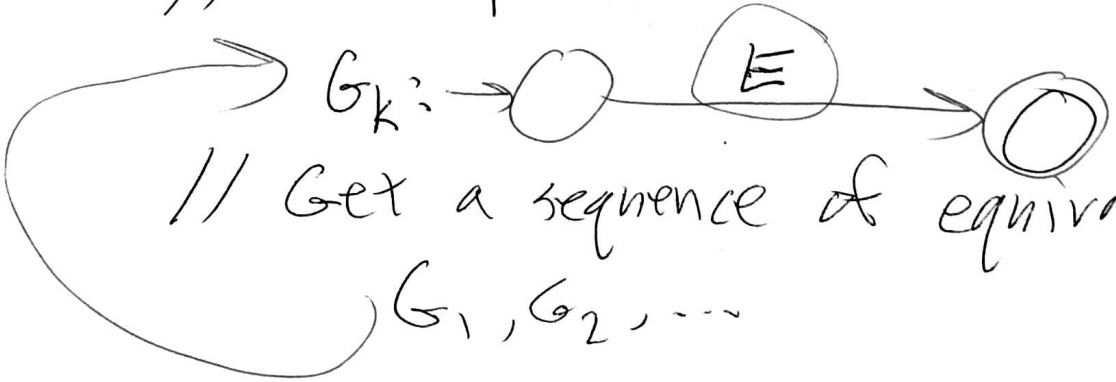
G accepts w if there exists a complete computation path of G on w ending in an accepting state of G.

Converting an ε-NFA to a regex. [State elimination method]

Given an ε-NFA N:

- Make N clean (if not already)
- Convert to an equivalent GNFA G_0
  (with same transition diagram, where $\varepsilon := \emptyset^*$)

// remove intermediate states one at
// a time until only the start state
// and accept states are left.



// Get a sequence of equivalent GNFA,
$G_1, G_2, \ldots$

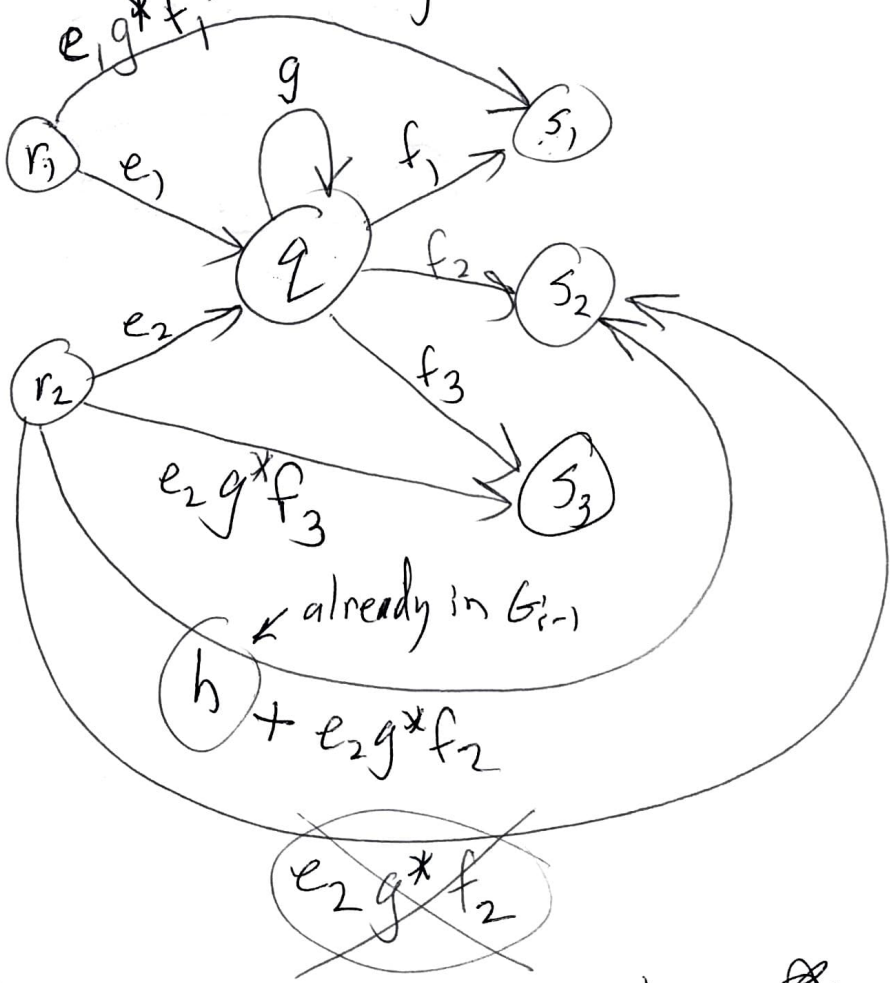
- White G_{i-1}

④

- For $i := 1, 2, \ldots$ as long as G_{i-1} has an intermediate state (not the start state & not the accept state)

Assume G_{i-1} is already constructed.

Construct G_i by choosing an intermediate state q of G_{i-1}, removing it and add bypassing arrows as follows:



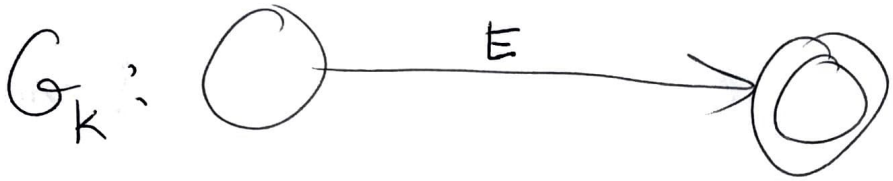
Do this for every combination of r_i & s_j

(5)

G_i is equivalent to $G_{(i-1)}$

endwhile

Finally left with no intermediate states.

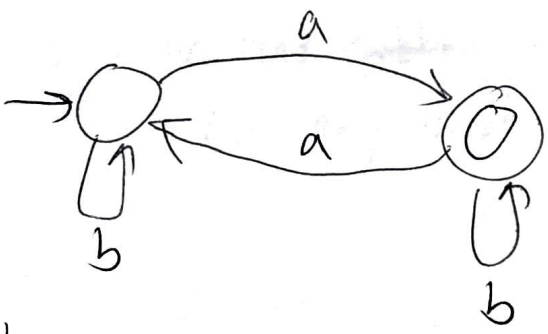


$L(G_k) = L(E)$

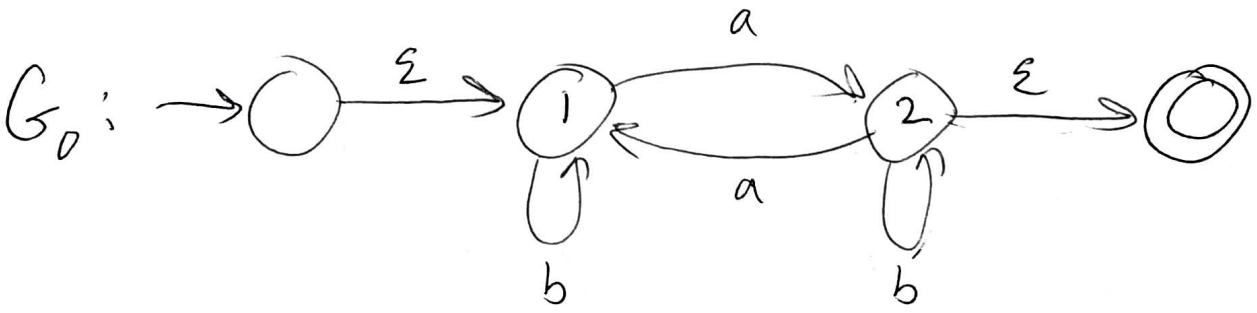
$\parallel$

$L(G_{k-1}) = L(G_{k-2}) = \cdots = L(G_0) = L(N)$.

EX: $\Sigma = \{a, b\}$ $L = \{w ;$ w has an odd # of a's$\}$

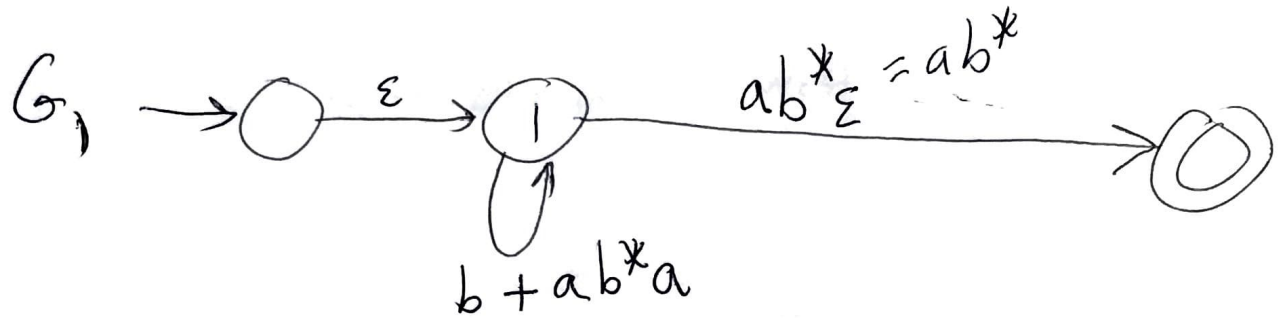


clean it up:

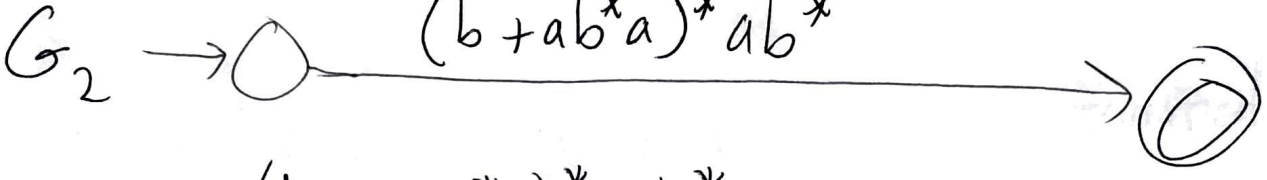


Eliminate state 2:

⑥

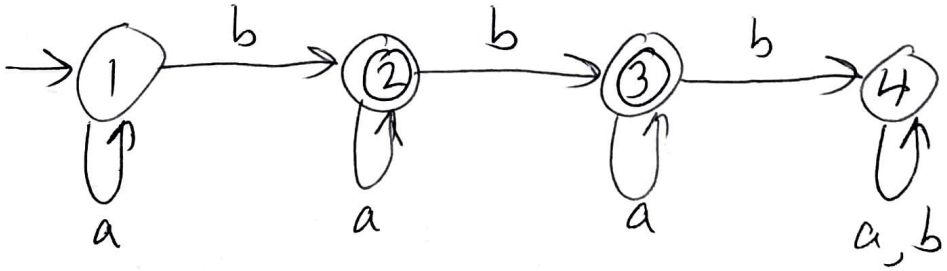


Elim 1:

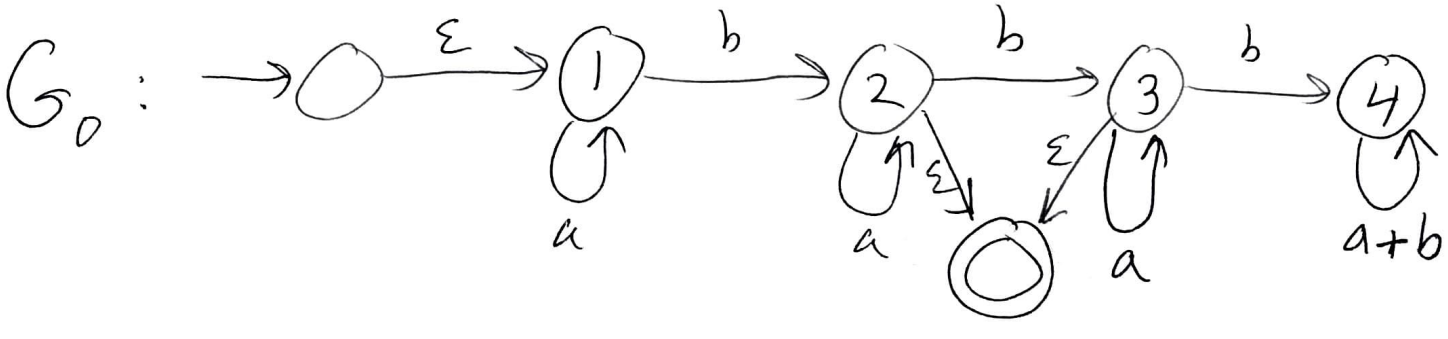


Answer: $\underbrace{(b + ab^*a)^*}_{\text{even \# of a's}} ab^*$

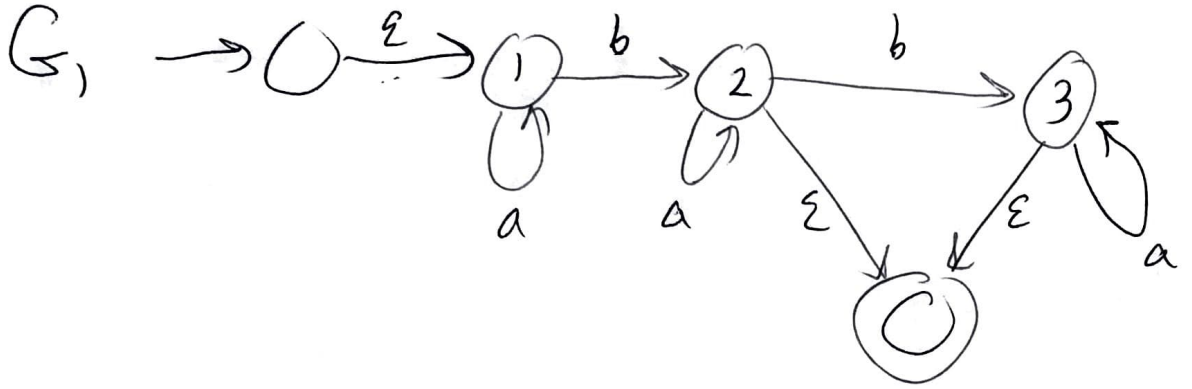
Ex: $L = \{w : w$ has ~~either~~ one or two b's (& any # of a's)$\}$



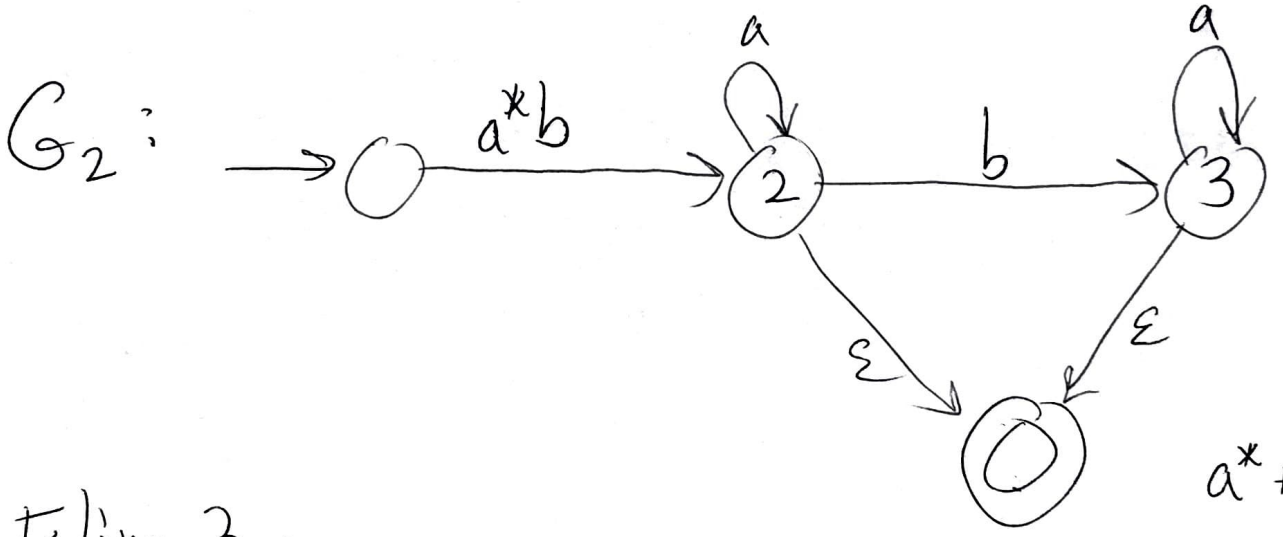
clean it up:



(7) Elim 4: (best choice — fewest bypasses (0))

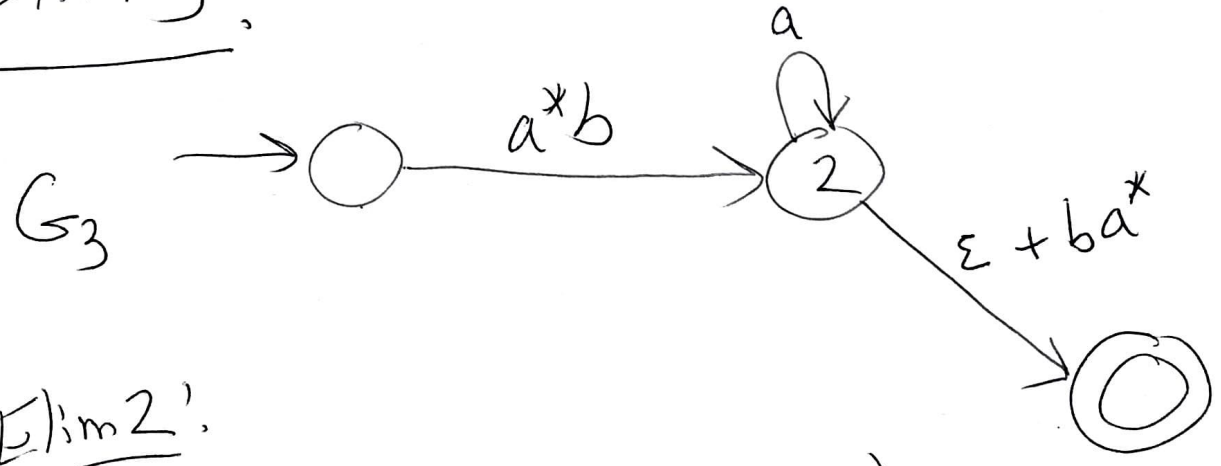


Elim 1 (tied with 3 for fewest required bypasses):



$a^* + \varepsilon \equiv a^*$

Elim 3:



Elim 2':