Automata Theory

CS411-2015F-06

Finite Automata & Regular Expressions

David Galles

Department of Computer Science
University of San Francisco

06-0: $L_{DFA} \& L_{REG}$

- $L_{DFA} = L_{NFA}$
- What about L_{REG}?
- How can we show that $L_{REG} = L_{DFA}$?

06-1: $L_{DFA} \& L_{REG}$

- $L_{DFA} = L_{NFA}$
- What about L_{REG}?
- How can we show that $L_{REG} = L_{DFA}$?
  - Show $L_{REG} \subseteq L_{NFA}$
  - Show $L_{NFA} \subseteq L_{REG}$

06-2: $L_{REG} \subseteq L_{NFA}$

- How can we show that $L_{REG} \subseteq L_{NFA}$?

06-3: $L_{REG} \subseteq L_{NFA}$

- How can we show that $L_{REG} \subseteq L_{NFA}$?
  - Given any regular expression r, create an NFA M such that $L[r] = L[M]$
  - Since regular expressions are defined recursively, our proof will be inductive
    - recursive $\approx$ inductive

06-4: $L_{REG} \subseteq L_{NFA}$

- To Prove: Given any regular expression r, we can create an NFA M such that $L[M] = L[r]$
  - $\exists$ NFA M s.t. $L[M] = L[r]$, $|F_M| = 1$, No transitions out of $f \in F$
- By induction on the structure of r

06-5: $L_{REG} \subseteq L_{NFA}$

Base Cases:

- r = a, a $\in \Sigma$

06-6: $L_{REG} \subseteq L_{NFA}$

Base Cases:

- r = a, a $\in \Sigma$

06-7: $L_{REG} \subseteq L_{NFA}$

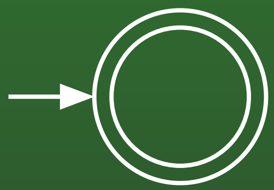
Base Cases:

- $r = \epsilon$

06-8: $L_{REG} \subseteq L_{NFA}$

Base Cases:

- $r = \epsilon$



06-9: $L_{REG} \subseteq L_{NFA}$

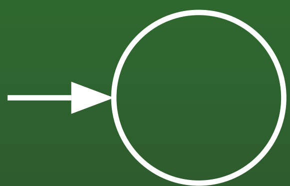
Base Cases:

- $r = \emptyset$

06-10: $L_{REG} \subseteq L_{NFA}$

Base Cases:

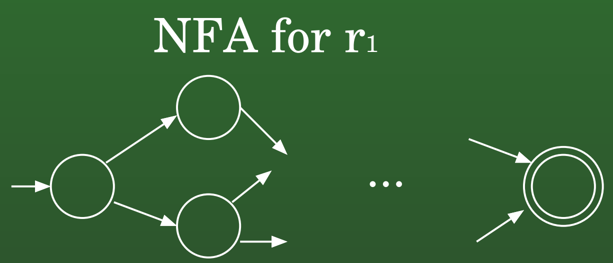
- $r = \emptyset$

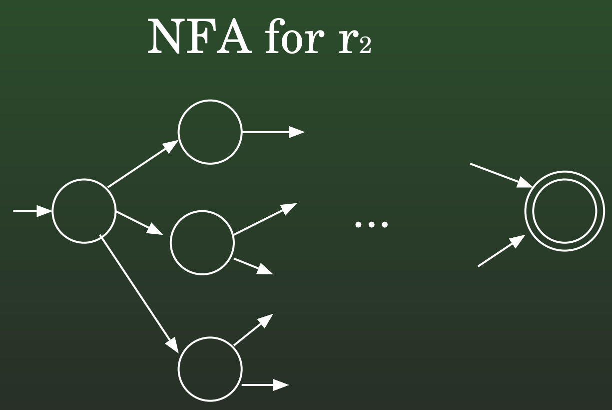


06-11: $L_{REG} \subseteq L_{NFA}$

Recursive Cases:

- $r = (r_1 r_2)$

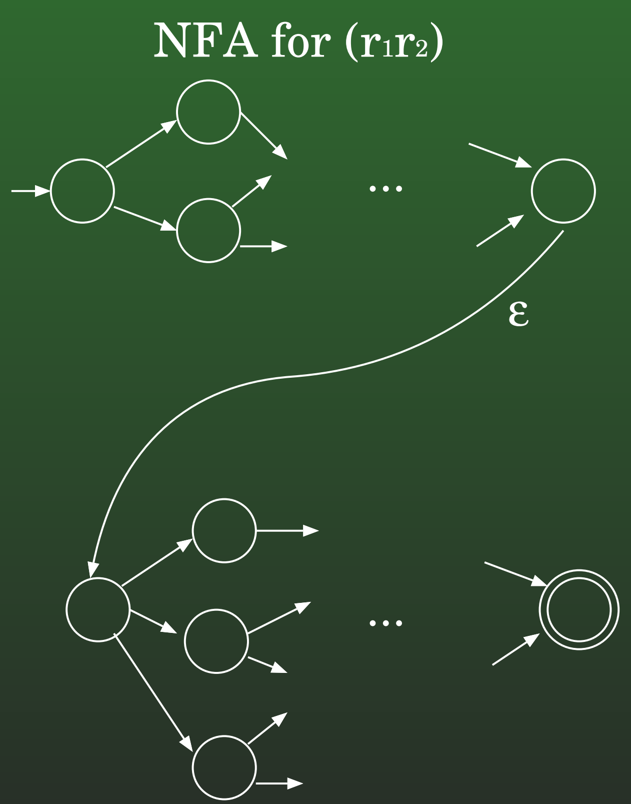




06-12: $L_{REG} \subseteq L_{NFA}$

Recursive Cases:

- $r = (r_1 r_2)$

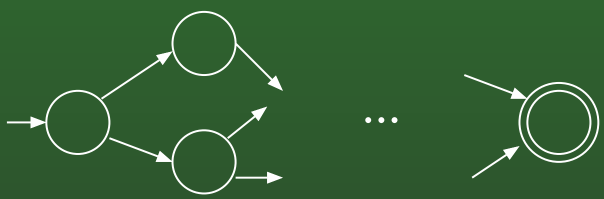


06-13: $L_{REG} \subseteq L_{NFA}$

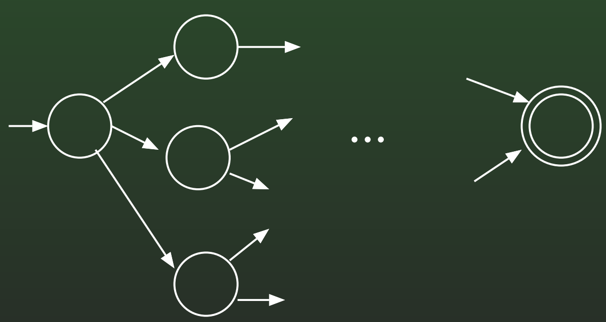
Recursive Cases:

- $r = (r_1 + r_2)$

NFA for r_1



NFA for r_2

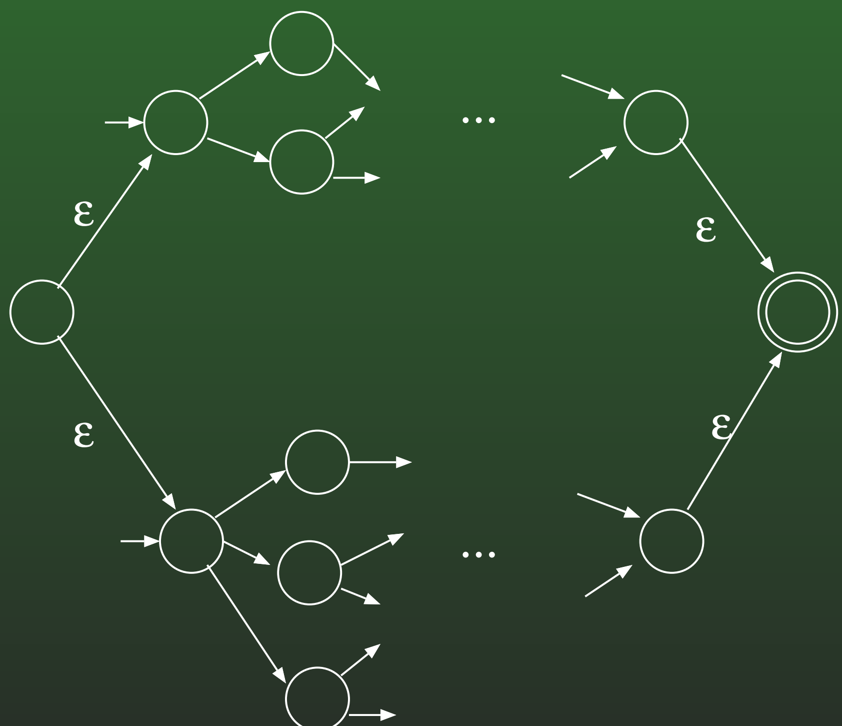


06-14: $L_{REG} \subseteq L_{NFA}$

Recursive Cases:

- $r = (r_1 + r_2)$

NFA for (r1+r2)

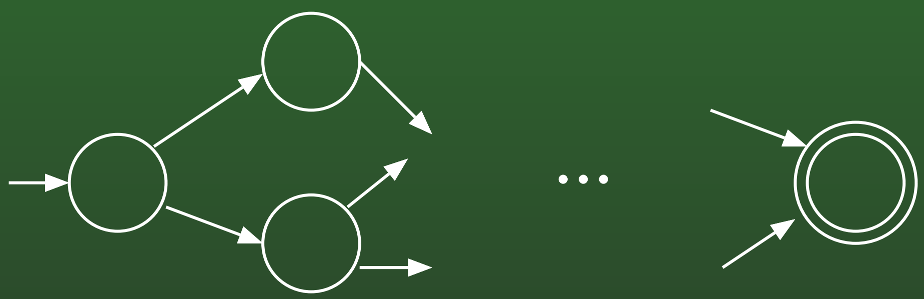


06-15: $L_{REG} \subseteq L_{NFA}$

Recursive Cases:

- $r = (r_1^*)$

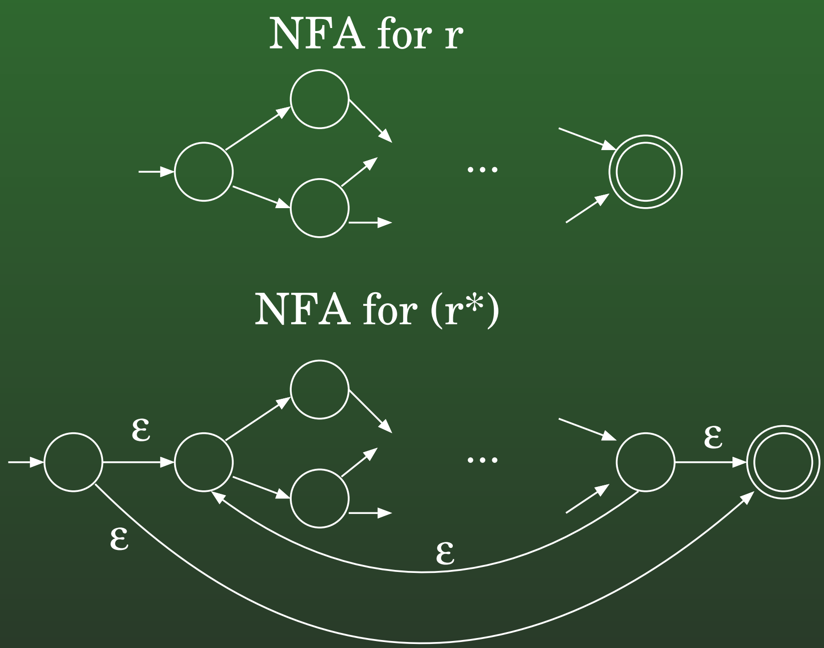
NFA for r



06-16: $L_{REG} \subseteq L_{NFA}$

Recursive Cases:

- $r = (r_1^*)$



06-17: $L_{REG} \subseteq L_{NFA}$

- Examples:
- 1(0+1)*0
  - ((1((0+1)*))0)

06-18: $L_{REG} \subseteq L_{NFA}$

- Examples:
- (a+b)*aba(a+b)*
  - ((((((a+b)*)a)b)a)((a+b)*))

06-19: $L_{REG} \subseteq L_{NFA}$

- Given any regular expresion r, we can create an NFA M such that $L[M] = L[r]$
- Given any NFA M, we can create a DFA M' such that $L[M'] = L[M]$
- Given any regular expresion r, we can create a DFA M such that $L[M] = L[r]$
- What about the other direction?

06-20: $L_{NFA} \subseteq L_{REG}$

- Start with a specialized NFA
  - No transitions into the start state
  - Single final state
  - No transitions out of the final state
- Can we transform any NFA into one in this form? How?

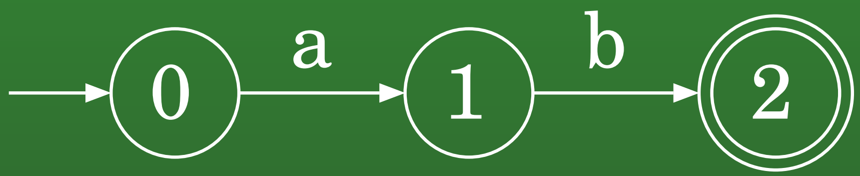
06-21: $L_{NFA} \subseteq L_{REG}$

- Transitions will be labeled with regular expressions
- If there is a transition from state q_1 to state q_2 labeled with regular expression r, then any string generated by r can move the machine from q_1 to q_2
  - Recall that $\forall$a $\in \Sigma$, a is a regular expression
  - Technically true, even for standard NFA

06-22: $L_{NFA} \subseteq L_{REG}$

- Transitions will be labeled with regular expressions
- If there is a transition from state q_1 to state q_2 labeled with regular expression r, then any string generated by r can move the machine from q_1 to q_2
  - Recall that $\forall$a $\in \Sigma$, a is a regular expression
  - Technically true, even for standard NFA
- Remove states, relabeling transitions so that the langauge defined by the machine does not change

06-23: $L_{NFA} \subseteq L_{REG}$

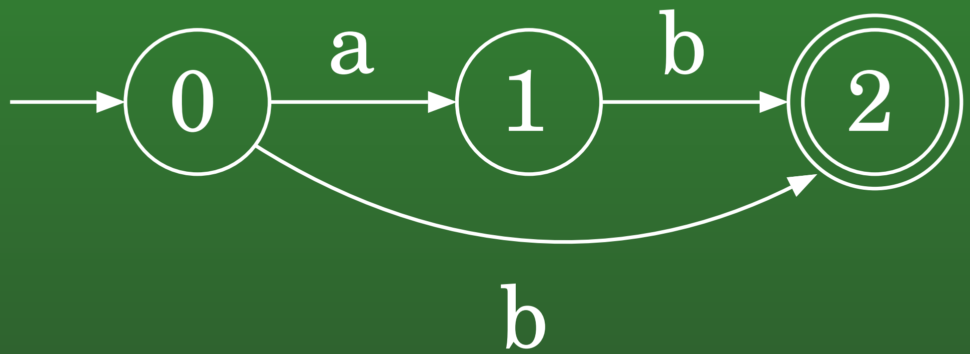


- Removing state q_1

06-24: $L_{NFA} \subseteq L_{REG}$

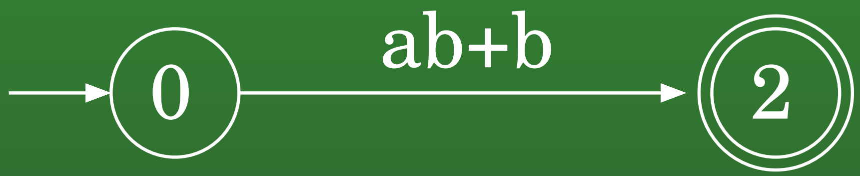
- State q_1 removed

06-25: $L_{NFA} \subseteq L_{REG}$



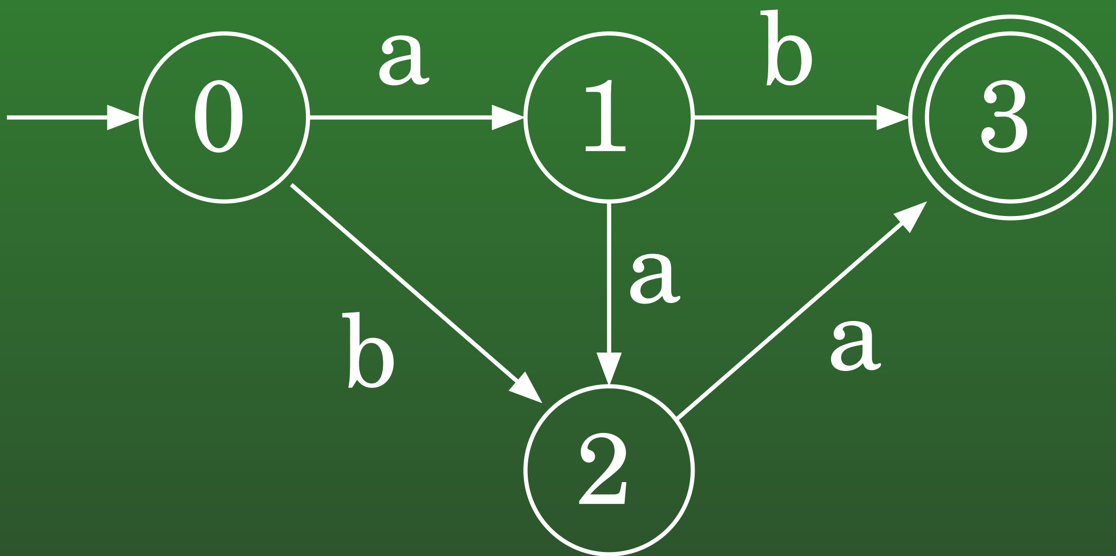
- Removing state q_1

06-26: $L_{NFA} \subseteq L_{REG}$



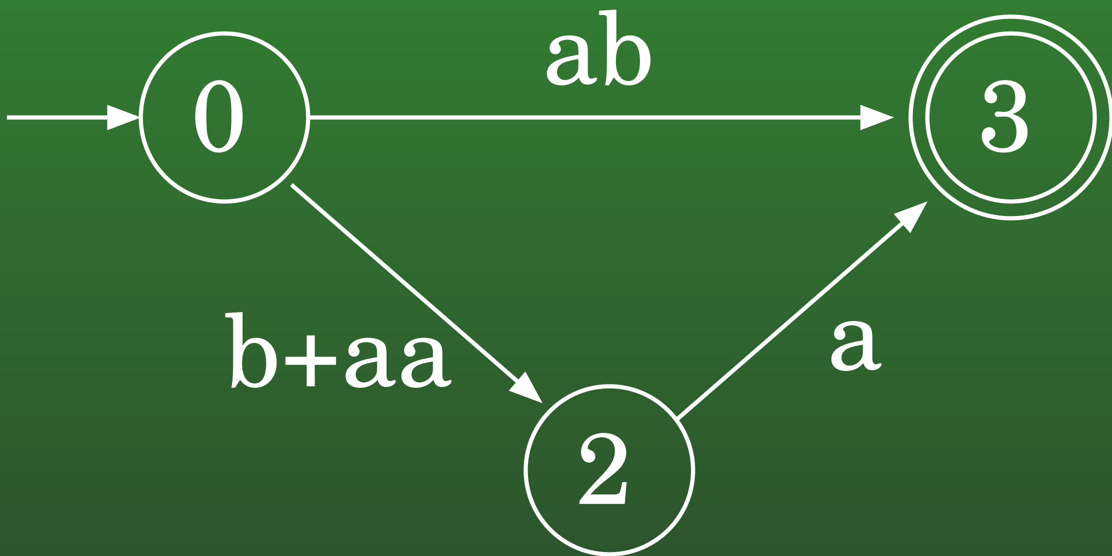
- State q_1 removed

06-27: $L_{NFA} \subseteq L_{REG}$



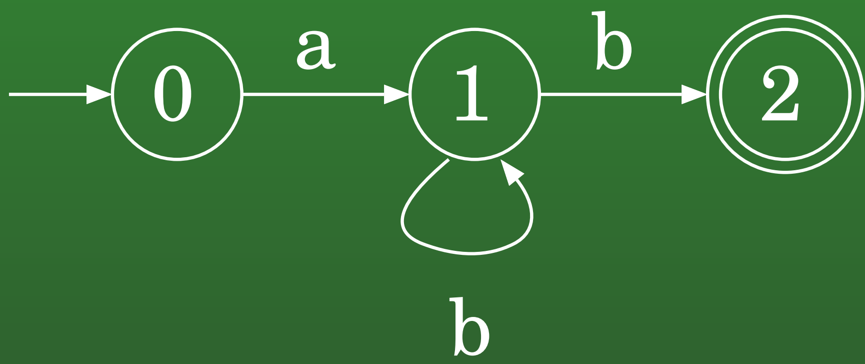
- Removing state q_1

06-28: $L_{NFA} \subseteq L_{REG}$



- State q_1 removed

06-29: $L_{NFA} \subseteq L_{REG}$

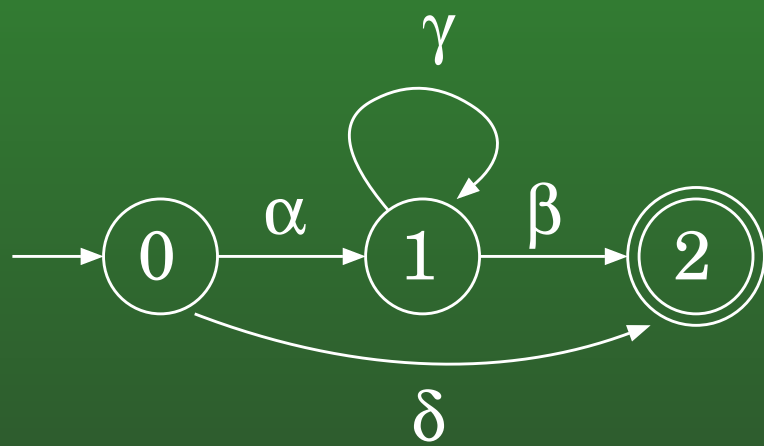


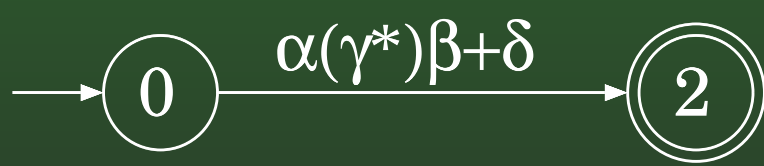
- Removing state q_1

06-30: $L_{NFA} \subseteq L_{REG}$

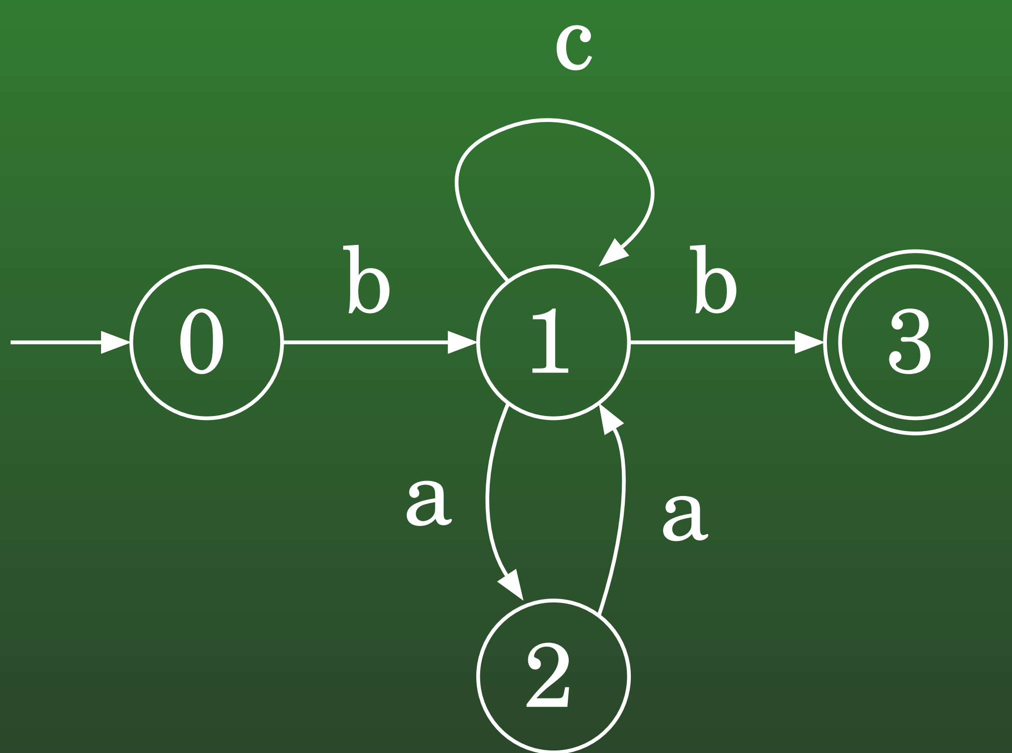
- Removing state q_1

06-31: $L_{NFA} \subseteq L_{REG}$



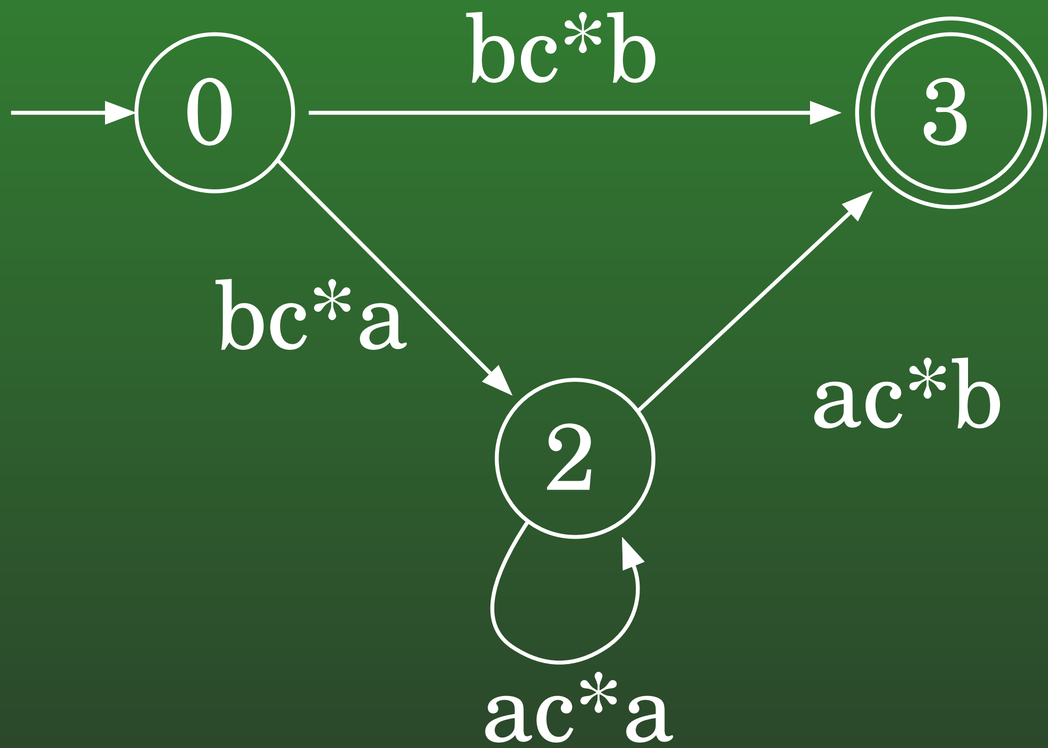


06-32: $L_{NFA} \subseteq L_{REG}$



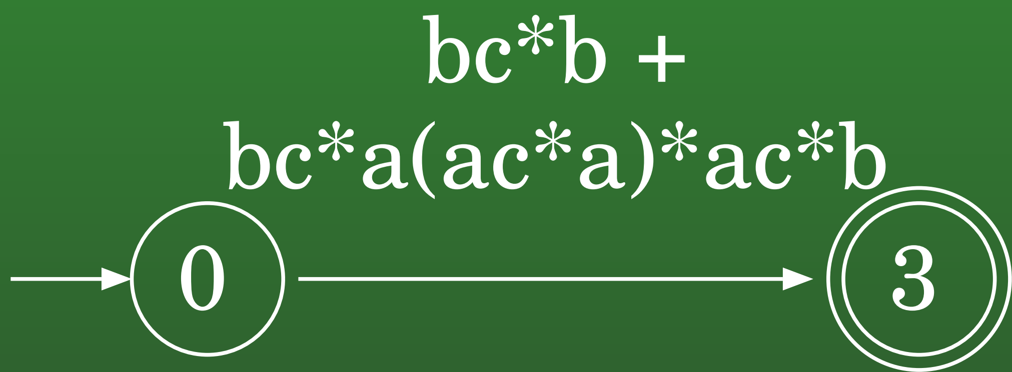
- Removing state q_1

06-33: $L_{NFA} \subseteq L_{REG}$



- State q_1 removed. Removing state q_2

 $L_{NFA} \subseteq L_{REG}$

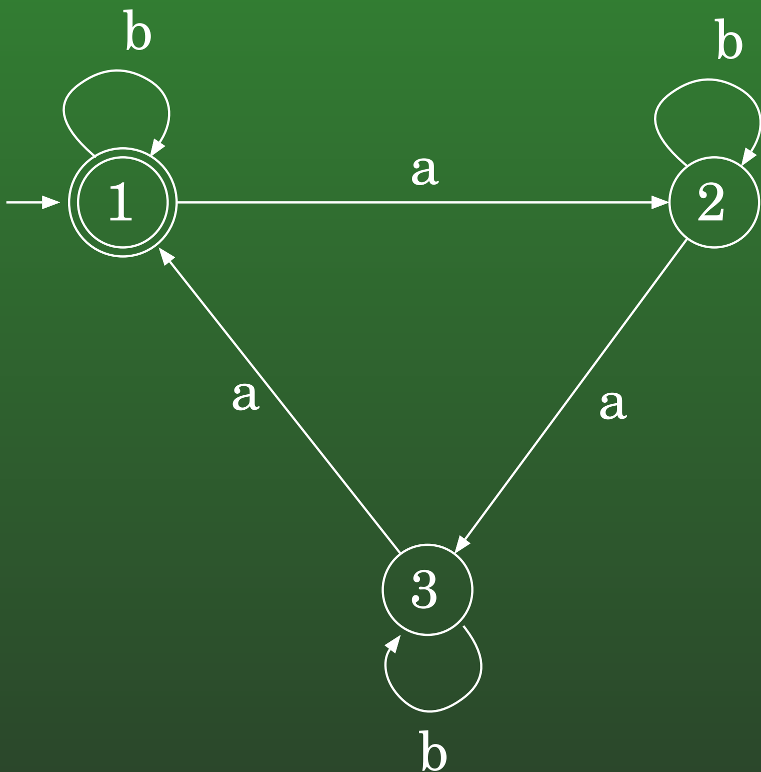


- State q_2 removed.

06-35: $L_{NFA} \subseteq L_{REG}$

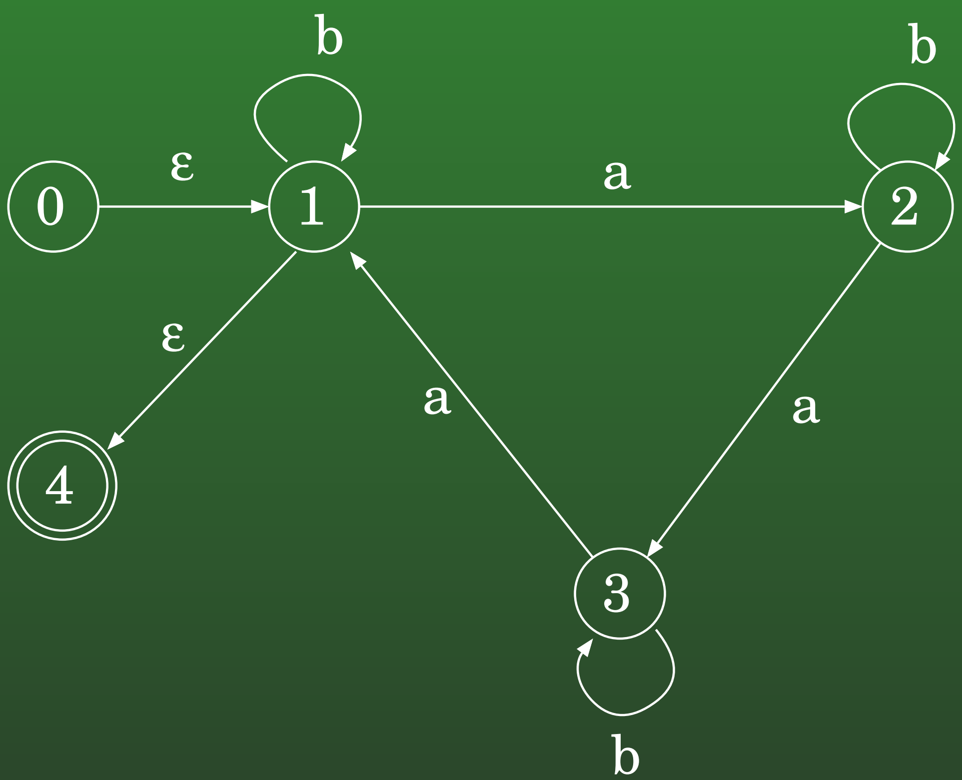
- Example:
  - NFA for all strings over {a,b} where # of a's mod 3 = 0

06-36: $L_{NFA} \subseteq L_{REG}$



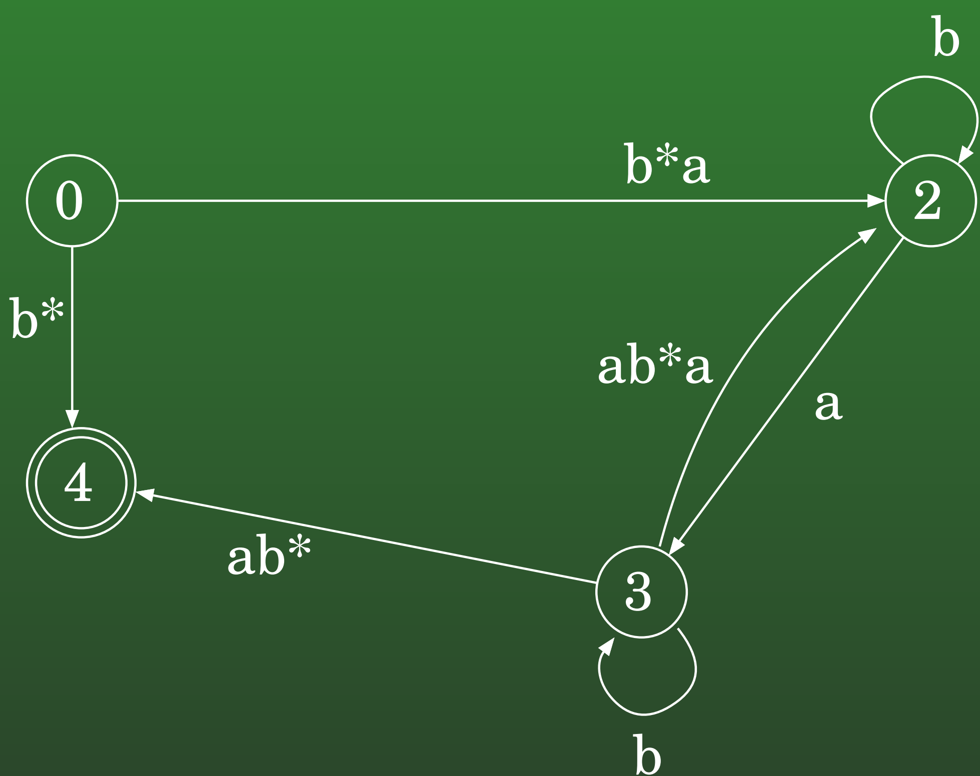
- Reconfigure NFA

06-37: $L_{NFA} \subseteq L_{REG}$



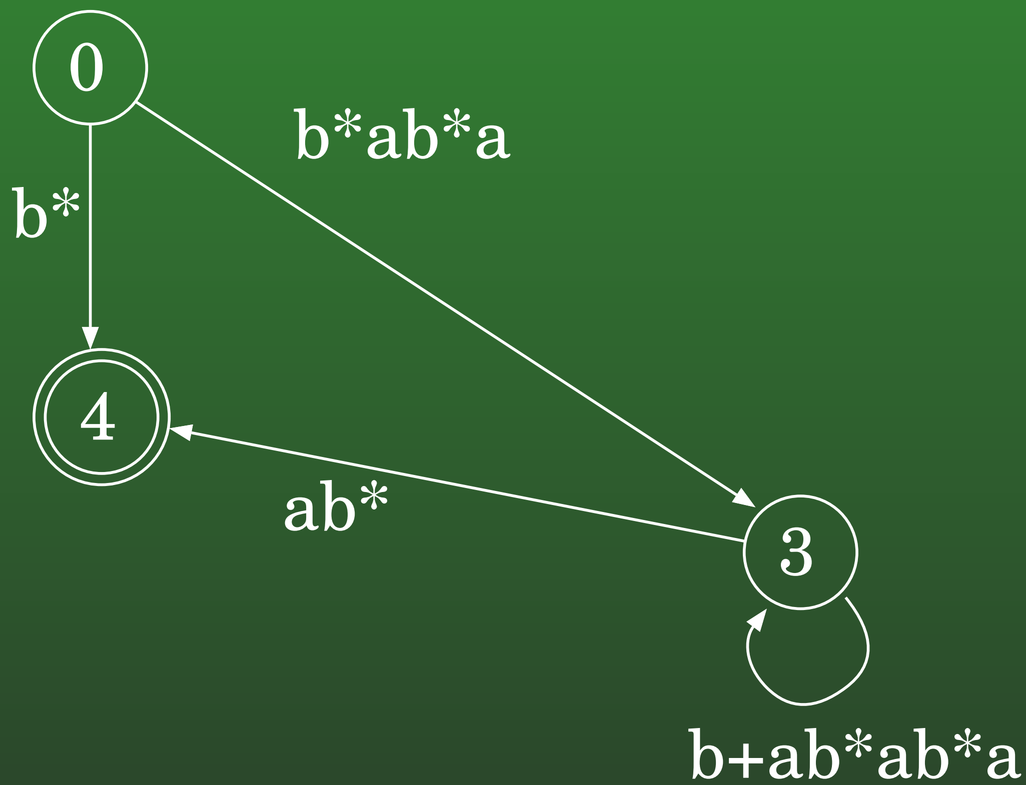
- Remove state q_1

06-38: $L_{NFA} \subseteq L_{REG}$



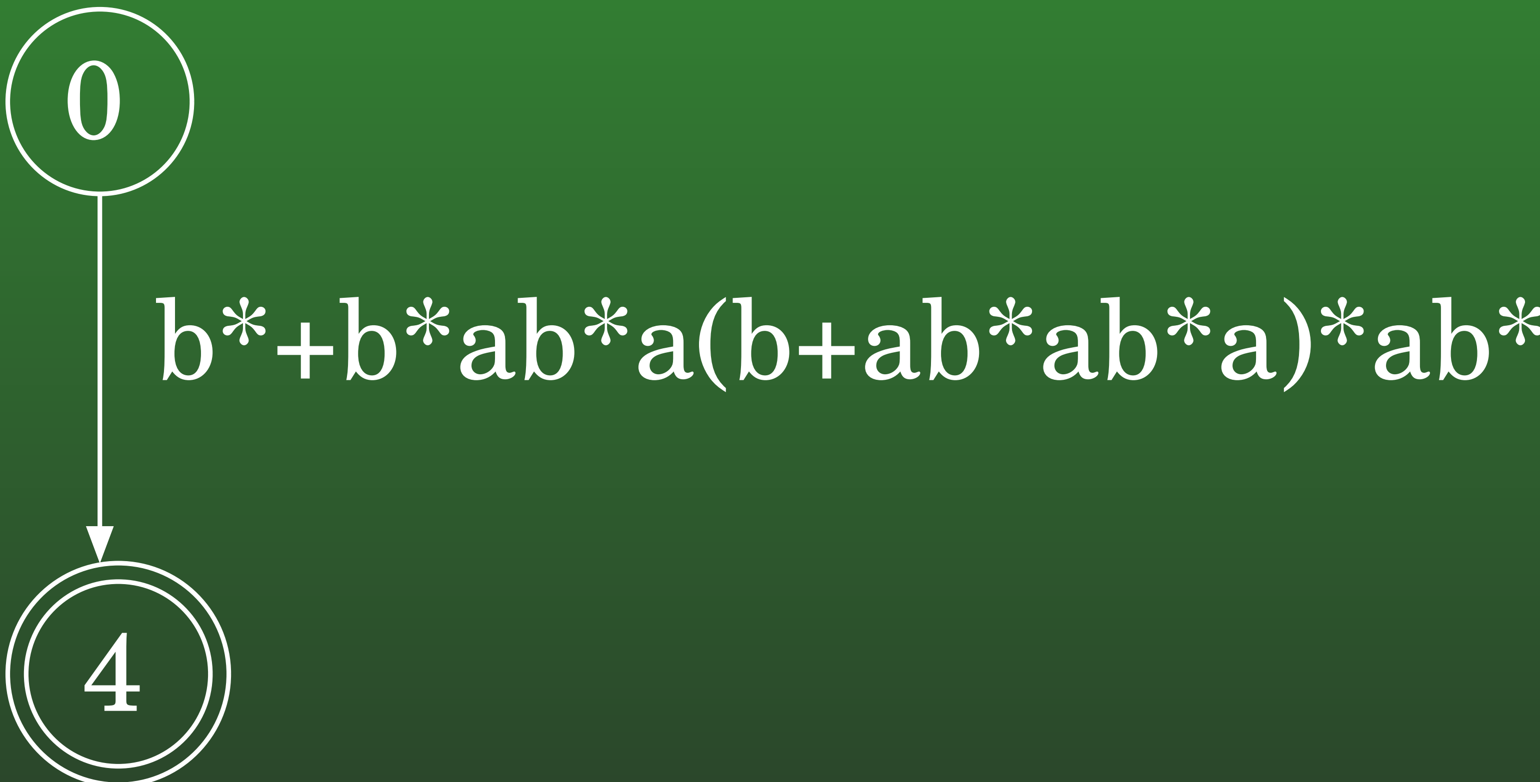
- State q_1 removed, removing state q_2

06-39: $L_{NFA} \subseteq L_{REG}$



- State q_2 removed, removing state q_3

06-40: $L_{NFA} \subseteq L_{REG}$



- State q_3 removed.