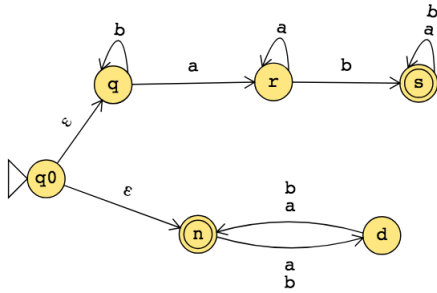


# Monday

The state diagram of an NFA over  $\{a, b\}$  is below. The formal definition of this NFA is:



Extra practice:  
Reconstruct

The language recognized by this NFA is:

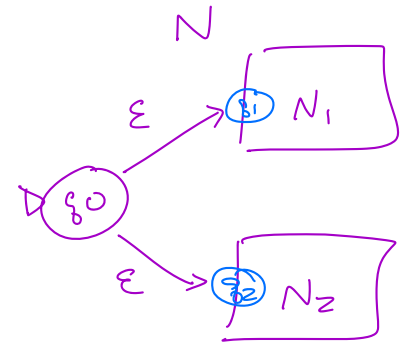
Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N_1$  such that  $L(N_1) = A_1$  and NFA  $N_2$  such that  $L(N_2) = A_2$ , then there is another NFA, let's call it  $N$ , such that  $L(N) = A_1 \cup A_2$ .

**Proof idea:** Use nondeterminism to choose which of  $N_1, N_2$  to run.

**Formal construction:** Let  $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$  and  $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$  and assume  $Q_1 \cap Q_2 = \emptyset$  and that  $q_0 \notin Q_1 \cup Q_2$ . Construct  $N = (Q, \Sigma, \delta, q_0, F_1 \cup F_2)$  where

- $Q = \{q_0\} \cup Q_1 \cup Q_2$
- $\delta : Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$  is defined by, for  $q \in Q$  and  $a \in \Sigma_\epsilon$ :

$$\delta(q, a) = \begin{cases} \{q_1, q_2\} & a = \epsilon, q = q_0 \\ \emptyset & a \in \Sigma, q = q_0 \\ \delta_1(q, a) & q \in Q_1, a \in \Sigma_\epsilon \\ \delta_2(q, a) & q \in Q_2, a \in \Sigma_\epsilon \end{cases}$$



*Proof of correctness would prove that  $L(N) = A_1 \cup A_2$  by considering an arbitrary string accepted by  $N$ , tracing an accepting computation of  $N$  on it, and using that trace to prove the string is in at least one of  $A_1, A_2$ ; then, taking an arbitrary string in  $A_1 \cup A_2$  and proving that it is accepted by  $N$ . Details left for extra practice.*

Over the alphabet  $\{a, b\}$ , the language  $L$  described by the regular expression  $\Sigma^* a \Sigma^* b$

includes the strings

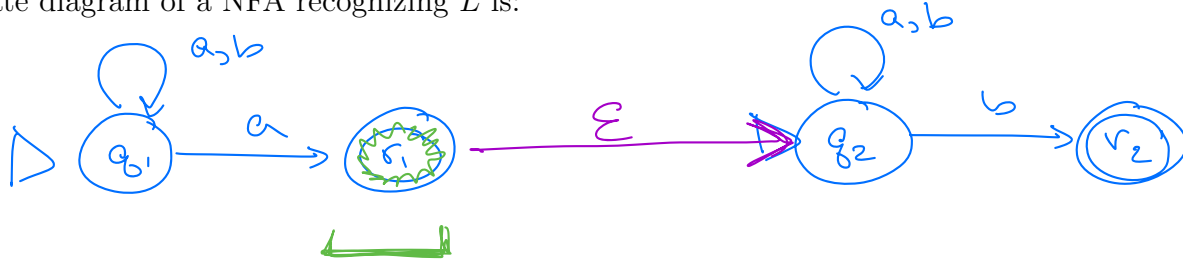
$ab$

and excludes the strings

$bbbb$

$\underbrace{\Sigma^* a}_{\text{first}} \cdot \Sigma^* b$

The state diagram of a NFA recognizing  $L$  is:



Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N_1$  such that  $L(N_1) = A_1$  and NFA  $N_2$  such that  $L(N_2) = A_2$ , then there is another NFA, let's call it  $N$ , such that  $L(N) = A_1 \circ A_2$ .

**Proof idea:** Allow computation to move between  $N_1$  and  $N_2$  "spontaneously" when reach an accepting state of  $N_1$ , guessing that we've reached the point where the two parts of the string in the set-wise concatenation are glued together.

**Formal construction:** Let  $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$  and  $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$  and assume  $Q_1 \cap Q_2 = \emptyset$ . Construct  $N = (Q, \Sigma, \delta, q_0, F)$  where



- $Q = Q_1 \cup Q_2$

- $q_0 = q_1$

- $F = F_2$

- $\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$  is defined by, for  $q \in Q$  and  $a \in \Sigma_\varepsilon$ :

$$\delta((q, a)) = \begin{cases} \delta_1((q, a)) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1((q, a)) & \text{if } q \in \underline{F_1} \text{ and } a \in \Sigma \\ \delta_1((q, a)) \cup \{q_2\} & \text{if } q \in \underline{F_1} \text{ and } a = \varepsilon \\ \delta_2((q, a)) & \text{if } q \in \bar{Q}_2 \end{cases}$$

if  $q \in Q_1$  and  $q \notin F_1$

if  $q \in \underline{F_1}$  and  $a \in \Sigma$

if  $q \in \underline{F_1}$  and  $a = \varepsilon$

if  $q \in \bar{Q}_2$

while reading first  
chars, simulate  
 $N_1$   
keep simulating  $N_1$   
spontaneously  
move to  $N_2$   
simulate  $N_2$  on  
the rest of input string.

don't lose  
spontaneous  
arrows within  
 $N_1$

spontaneous  
move to  $N_2$

*Proof of correctness would prove that  $L(N) = A_1 \circ A_2$  by considering an arbitrary string accepted by  $N$ , tracing an accepting computation of  $N$  on it, and using that trace to prove the string can be written as the result of concatenating two strings, the first in  $A_1$  and the second in  $A_2$ ; then, taking an arbitrary string in  $A_1 \circ A_2$  and proving that it is accepted by  $N$ . Details left for extra practice.*

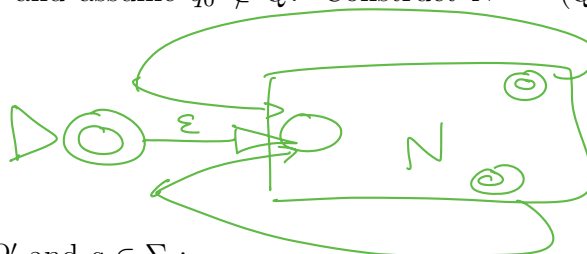
Suppose  $A$  is a language over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N$  such that  $L(N) = A$ , then there is another NFA, let's call it  $N'$ , such that  $L(N') = A^*$ .

**Proof idea:** Add a fresh start state, which is an accept state. Add spontaneous moves from each (old) accept state to the old start state.

**Formal construction:** Let  $N = (Q, \Sigma, \delta, q_1, F)$  and assume  $q_0 \notin Q$ . Construct  $N' = (Q', \Sigma, \delta', q_0, F')$  where

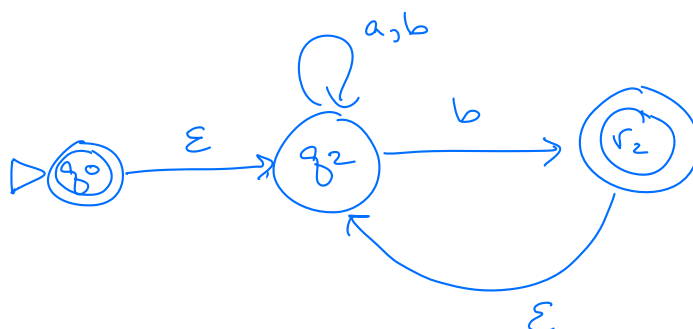
- $Q' = Q \cup \{q_0\}$
- $F' = F \cup \{q_0\}$
- $\delta' : Q' \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q')$  is defined by, for  $q \in Q'$  and  $a \in \Sigma_\epsilon$ :

$$\delta'((q, a)) = \begin{cases} \delta((q, a)) & \text{if } q \in Q \text{ and } q \notin F \\ \delta((q, a)) & \text{if } q \in F \text{ and } a \in \Sigma \\ \delta((q, a)) \cup \{q_1\} & \text{if } q \in F \text{ and } a = \epsilon \\ \{q_1\} & \text{if } q = q_0 \text{ and } a = \epsilon \\ \emptyset & \text{if } q = q_0 \text{ and } a \in \Sigma \end{cases}$$



*Proof of correctness would prove that  $L(N') = A^*$  by considering an arbitrary string accepted by  $N'$ , tracing an accepting computation of  $N'$  on it, and using that trace to prove the string can be written as the result of concatenating some number of strings, each of which is in  $A$ ; then, taking an arbitrary string in  $A^*$  and proving that it is accepted by  $N'$ . Details left for extra practice.*

**Application:** A state diagram for a NFA over  $\Sigma = \{a, b\}$  that recognizes  $L((\Sigma^*b)^*)$ :



Transition function  
DFA  $\delta: Q \times \Sigma \rightarrow Q$   
NFA  $\delta: Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$

Review Quiz

**True or False:** The state diagram of any DFA is also the state diagram of a NFA.

**True or False:** The state diagram of any NFA is also the state diagram of a DFA.

**True or False:** The formal definition  $(Q, \Sigma, \delta, q_0, F)$  of any DFA is also the formal definition of a NFA.

**True or False:** The formal definition  $(Q, \Sigma, \delta, q_0, F)$  of any NFA is also the formal definition of a DFA.

## Review: Week 3 Monday

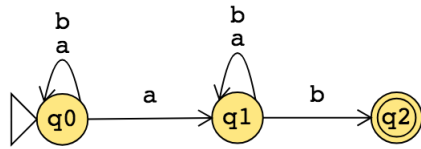
Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes each week are all due on Friday (with no penalty late submission open until Sunday).

Please complete the review quiz questions on [Gradescope](#) about constructions using NFAs.

**Pre class reading for next time:** Theorem 1.39 “Proof Idea”, Example 1.41, Example 1.56, Example 1.58.

# Wednesday

Consider the state diagram of an NFA over  $\{a, b\}$ :



Example strings accepted  
 $ab$   
 $aaa b b b a b$   
 $ab b$

Example strings rejected  
 $\epsilon$      $a$      $b$      $aba$

$\{w \in \{a, b\}^* \mid w \text{ has } ab \text{ as a substring and ends in } b\}$

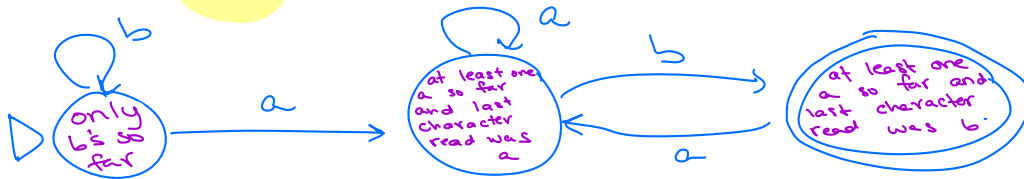
$\{w \in \{a, b\}^* \mid w \text{ has at least one } a \text{ and ends in } b\}$

$L((a^*b^*)^* a (a^*b^*)^* b)$

$L((a \cup b)^* a (a \cup b)^* b)$

The language recognized by this NFA is

The state diagram of a DFA recognizing this same language is:



Suppose  $A$  is a language over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N$  such that  $L(N) = A$  then there is a DFA  $M$  such that  $L(M) = A$ .

**Proof idea:** States in  $M$  are "macro-states" – collections of states from  $N$  – that represent the set of possible states a computation of  $N$  might be in.

**Formal construction:** Let  $N = (Q, \Sigma, \delta, q_0, F)$ . Define

NFA

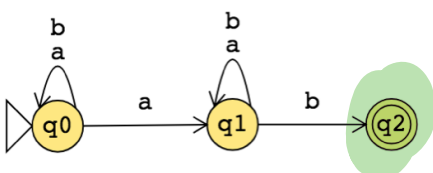
$M = (\mathcal{P}(Q), \Sigma, \delta', q', \{X \subseteq Q \mid X \cap F \neq \emptyset\})$

DFA

where  $q' = \{q \in Q \mid q = q_0 \text{ or is accessible from } q_0 \text{ by spontaneous moves in } N\}$  and

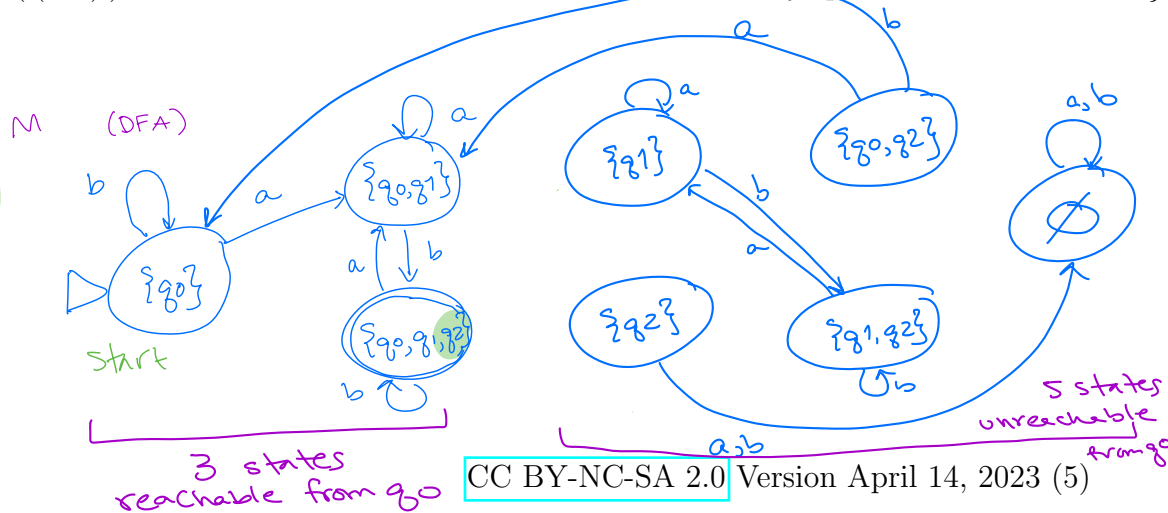
$\delta'((X, x)) = \{q \in Q \mid q \in \delta(r, x) \text{ for some } r \in X \text{ or is accessible from such an } r \text{ by spontaneous moves in } N\}$

$\Sigma = \{a, b\}$

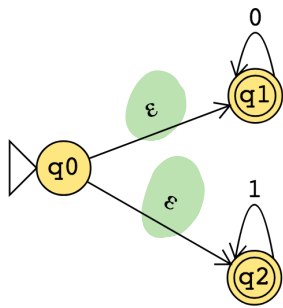


$N$

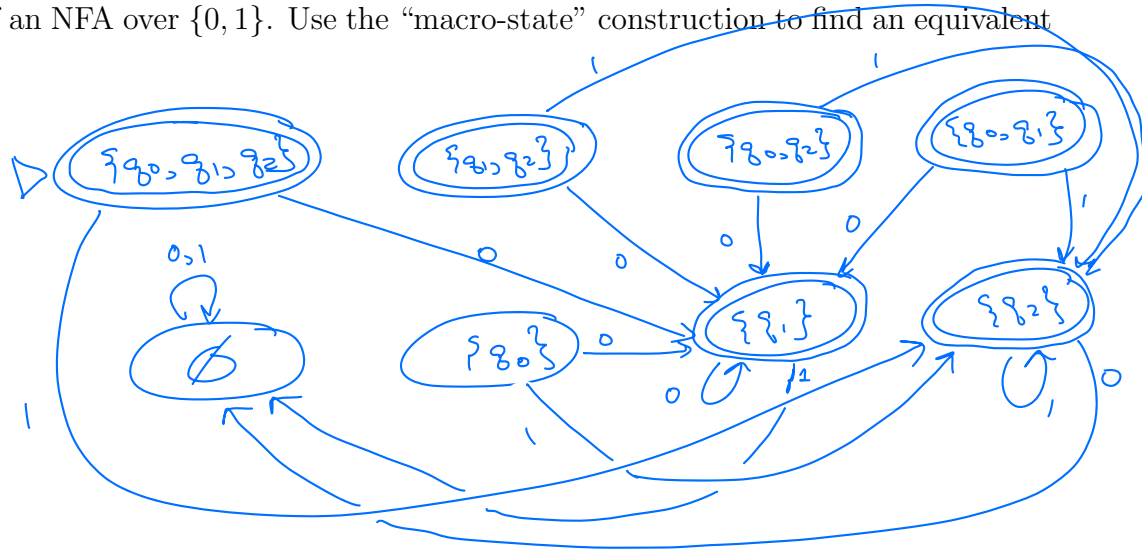
3 states  
 $|\mathcal{P}(Q)| = 2^3 = 8$



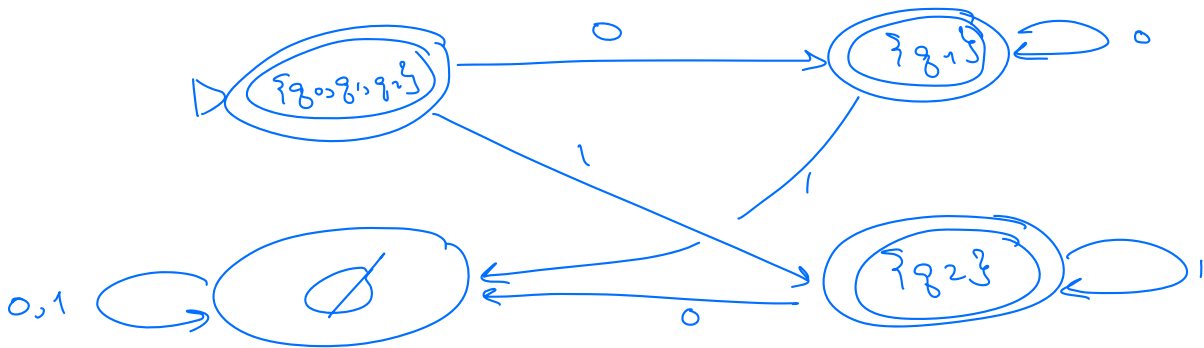
Consider the state diagram of an NFA over  $\{0, 1\}$ . Use the “macro-state” construction to find an equivalent DFA.



3 states in NFA  
 $2^3 = 8$



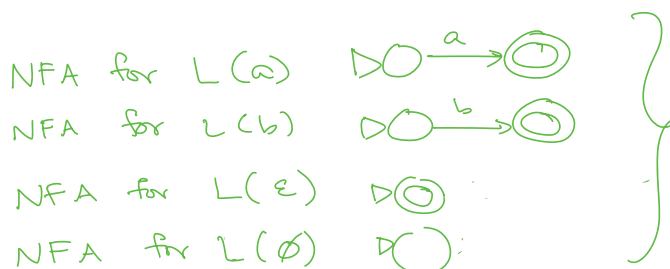
Prune this diagram to get an equivalent DFA with only the “macro-states” reachable from the start state.



Suppose  $A$  is a language over an alphabet  $\Sigma$ . **Claim:** if there is a regular expression  $R$  such that  $L(R) = A$ , then there is a NFA, let's call it  $N$ , such that  $L(N) = A$ .

**Structural induction:** Regular expression is built from basis regular expressions using inductive steps (union, concatenation, Kleene star symbols). Use constructions to mirror these in NFAs.

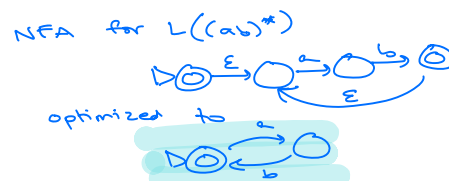
**Application:** A state diagram for a NFA over  $\{a, b\}$  that recognizes  $L(a^*(ab)^*)$ :



Put these together using constructions for  $\cup$ ,  $\circ$ ,  $*$ .



optimized to



$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}.$$

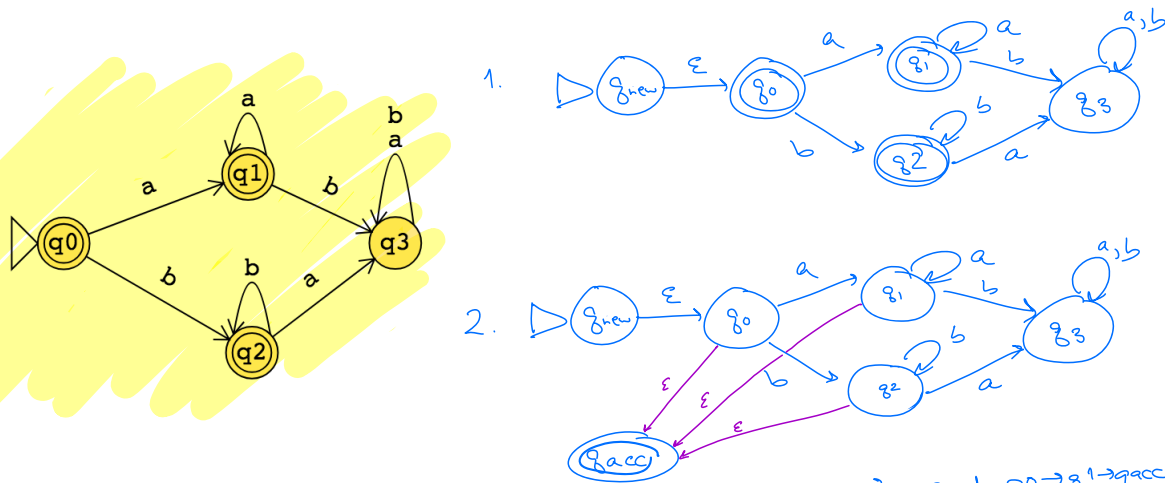
Suppose  $A$  is a language over an alphabet  $\Sigma$ . **Claim:** if there is a DFA  $M$  such that  $L(M) = A$ , then there is a regular expression, let's call it  $R$ , such that  $L(R) = A$ .

**Proof idea:** Trace all possible paths from start state to accept state. Express labels of these paths as regular expressions, and union them all.

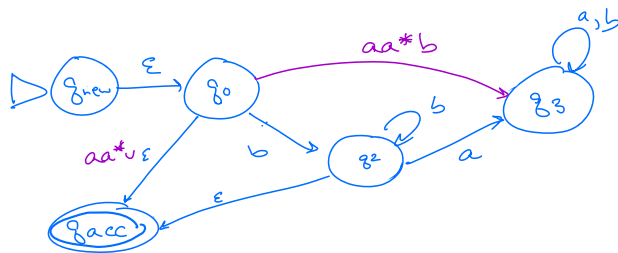
Intermediate diagram is called "GNFA"

1. Add new start state with  $\epsilon$  arrow to old start state.
2. Add new accept state with  $\epsilon$  arrow from old accept states. Make old accept states non-accept.
3. Remove one (of the old) states at a time: modify regular expressions on arrows that went through removed state to restore language recognized by machine.

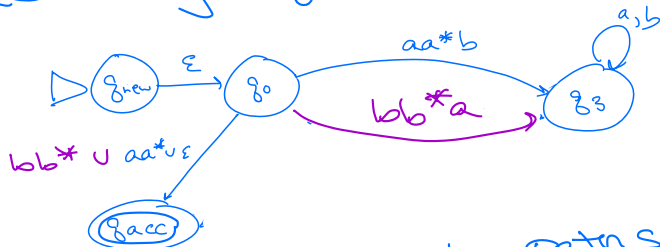
**Application:** Find a regular expression describing the language recognized by the DFA with state diagram



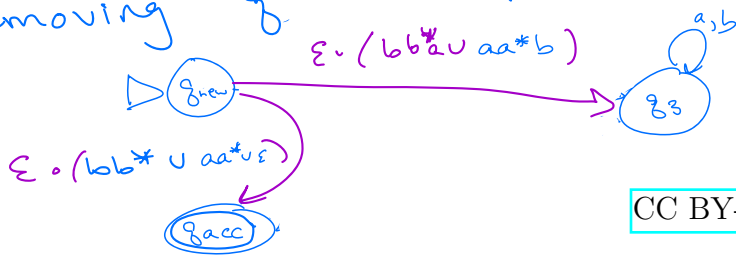
3. Removing  $q_1$ : note paths  $q_0 \rightarrow q_1 \rightarrow q_3$  and  $q_0 \rightarrow q_1 \rightarrow q_{acc}$



Removing  $q_2$ : note paths  $q_0 \rightarrow q_2 \rightarrow q_3$  and  $q_0 \rightarrow q_2 \rightarrow q_{acc}$



Removing  $q_0$ : note paths  $q_{new} \rightarrow q_0 \rightarrow q_3$  and  $q_{new} \rightarrow q_0 \rightarrow q_{acc}$



$R$  is  $\epsilon \cup (bb^* \cup aa^* \cup \epsilon)$

**Conclusion:** For each language  $L$ ,

There is a DFA that recognizes  $L \iff \exists M (M \text{ is a DFA and } L(M) = A)$   
if and only if

There is a NFA that recognizes  $L \iff \exists N (N \text{ is a NFA and } L(N) = A)$   
if and only if

There is a regular expression that describes  $L \iff \exists R (R \text{ is a regular expression and } L(R) = A)$

A language is called **regular** when any (hence all) of the above three conditions are met.

## Review: Week 3 Wednesday

Please complete the review quiz questions on [Gradescope](#) about translating between DFA, NFA, and regular expressions.

**Pre class reading for next time:** Introduction to Section 1.4 (page 77)



## Friday

**Theorem:** For an alphabet  $\Sigma$ , For each language  $L$  over  $\Sigma$ ,

$$\begin{array}{c} L \text{ is recognized by some DFA} \\ \text{iff} \\ L \text{ is recognized by some NFA} \\ \text{iff} \\ L \text{ is described by some regular expression} \end{array}$$

If (any, hence all) these conditions apply,  $L$  is called **regular**.

**Prove or Disprove:** There is some alphabet  $\Sigma$  for which there is some language recognized by an NFA but not by any DFA.

**Prove or Disprove:** There is some alphabet  $\Sigma$  for which there is some finite language not described by any regular expression over  $\Sigma$ .

**Prove or Disprove:** If a language is recognized by an NFA then the complement of this language is not recognized by any DFA.

Set	Cardinality
$\{0, 1\}$	
$\{0, 1\}^*$	
$\mathcal{P}(\{0, 1\})$	
The set of all languages over $\{0, 1\}$	
The set of all regular expressions over $\{0, 1\}$	
The set of all regular languages over $\{0, 1\}$	

**Pumping Lemma** (Sipser Theorem 1.70): If  $A$  is a regular language, then there is a number  $p$  (a *pumping length*) where, if  $s$  is any string in  $A$  of length at least  $p$ , then  $s$  may be divided into three pieces,  $s = xyz$  such that

- $|y| > 0$
- for each  $i \geq 0$ ,  $xy^iz \in A$
- $|xy| \leq p$ .

**True or False:** A pumping length for  $A = \{0, 1\}^*$  is  $p = 5$ .

**True or False:** A pumping length for  $A = \{1, 01, 001, 0001, 00001\}$  is  $p = 4$ .

**True or False:** A pumping length for  $A = \{0^j1 \mid j \geq 0\}$  is  $p = 3$ .

**True or False:** For any language  $A$ , if  $p$  is a pumping length for  $A$  and  $p' > p$ , then  $p'$  is also a pumping length for  $A$ .

## Review: Week 3 Friday

Please complete the review quiz questions on [Gradescope](#) about the class of regular languages.

**Pre class reading for next time:** Example 1.75, Example 1.77