

# Equivalence of DFA and Regular Expressions

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

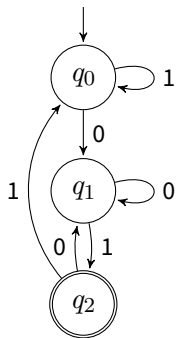
Chinese University of Hong Kong

Fall 2016

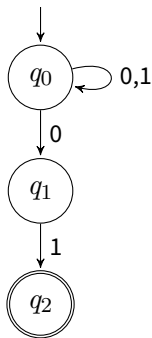
## Three ways of doing it

$$L = \{x \in \Sigma^* \mid x \text{ ends in } 01\}$$

$$\Sigma = \{0, 1\}$$



DFA

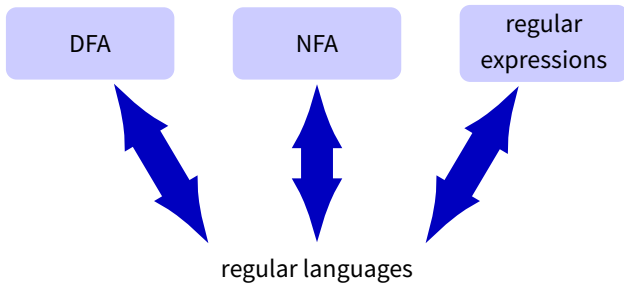


NFA

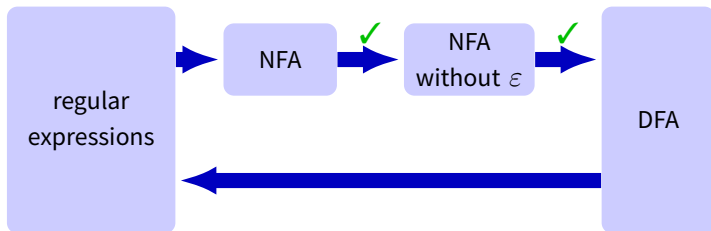
$(0 + 1)^*01$

regular  
expressions

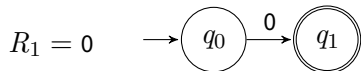
They are equally powerful



# Roadmap

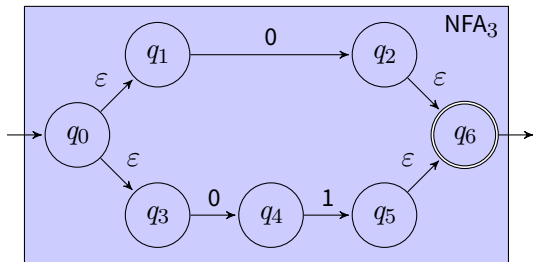


## Examples: regular expression $\rightarrow$ NFA

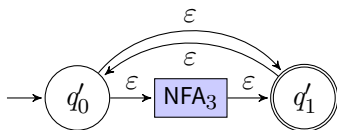


# Examples: regular expression $\rightarrow$ NFA

$$R_3 = 0 + 01$$



$$R_4 = (0 + 01)^*$$



# Regular expressions

In general, how do we convert a regular expression to an NFA?

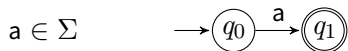
A **regular expression** over  $\Sigma$  is an expression formed by the following rules

- ▶ The symbols  $\emptyset$  and  $\varepsilon$  are regular expressions
- ▶ Every symbol  $a$  in  $\Sigma$  is a regular expression
- ▶ If  $R$  and  $S$  are regular expressions, so are  $R + S$ ,  $RS$  and  $R^*$

# General method

regular  
expression  $\implies$  NFA

---

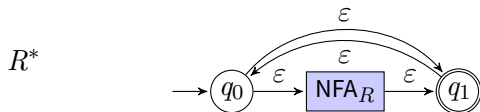
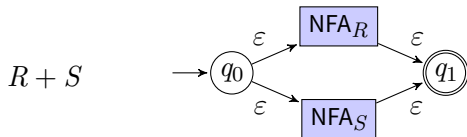
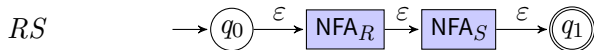




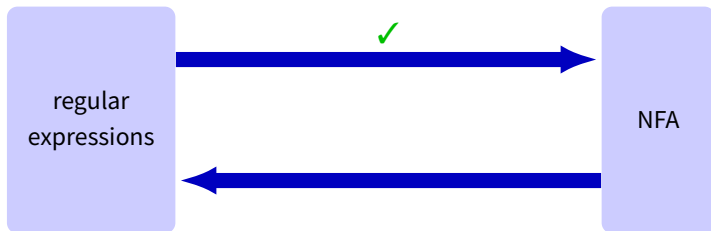
# General method

regular expression  $\implies$  NFA

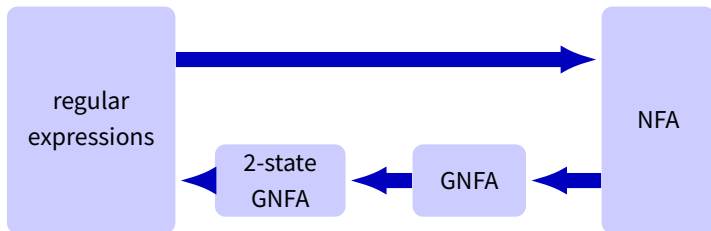
---



# Roadmap



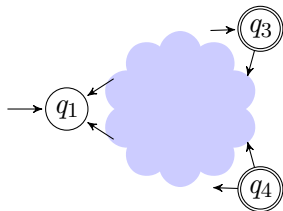
# Roadmap



## Simplify the NFA

First we simplify the NFA so that

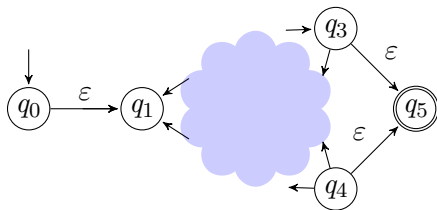
- ▶ It has **exactly one** accepting state
- ▶ No arrows come into the start state
- ▶ No arrows go out of the accepting state



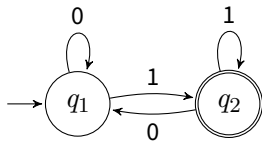
# Simplify the NFA

First we simplify the NFA so that

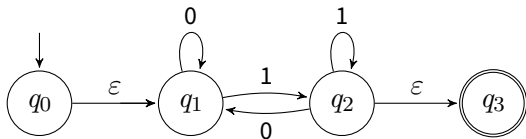
- ▶ It has **exactly one** accepting state
- ▶ No arrows come into the start state
- ▶ No arrows go out of the accepting state



## Simplify the NFA



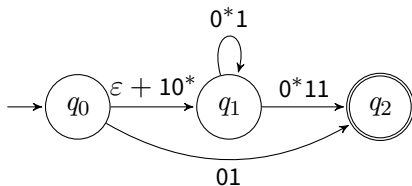
## Simplify the NFA



- ▶ It has **exactly one** accepting state ✓
- ▶ No arrows come into the start state ✓
- ▶ No arrows go out of the accepting state ✓

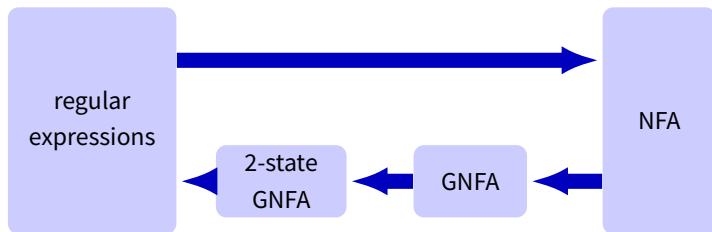
## Generalized NFAs

A **generalized NFA** is an NFA whose transitions are labeled by **regular expressions**, like



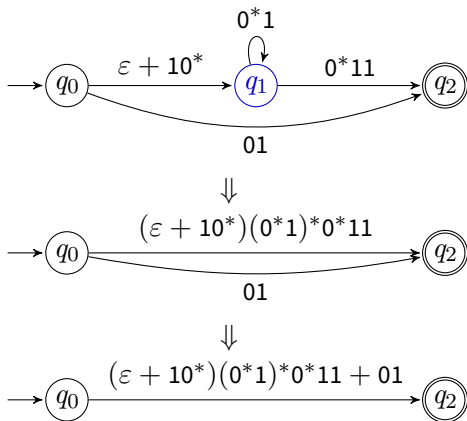


## GNFA state elimination



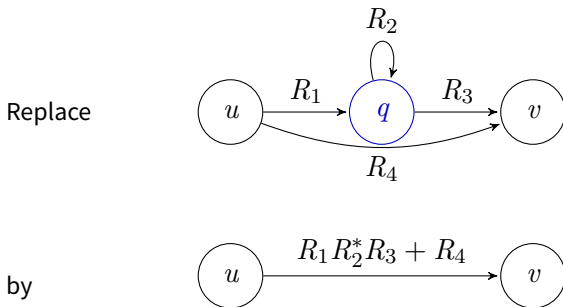
We will **eliminate** every state but the start and accepting states

# State elimination



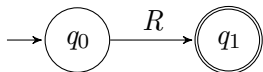
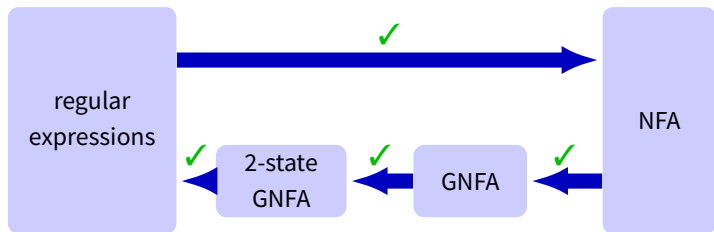
## State elimination: general method

To **eliminate** state  $q$ , for every pair of states  $(u, v)$



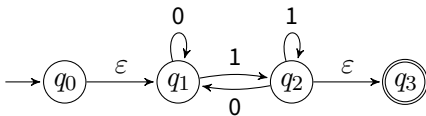
Remember to do this **even when**  $u = v$

# Roadmap



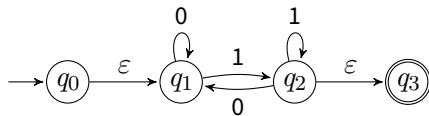
A 2-state GNFA is the same as a regular expression  $R$

## Conversion example

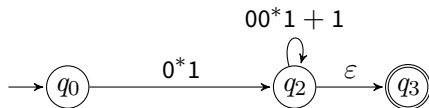


Eliminate  $q_1$ :

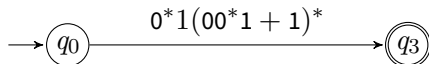
## Conversion example



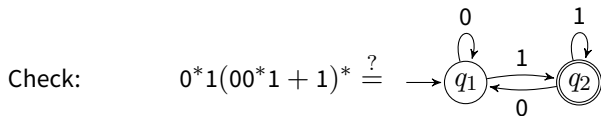
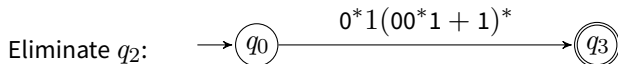
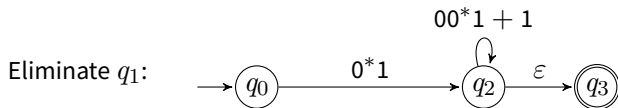
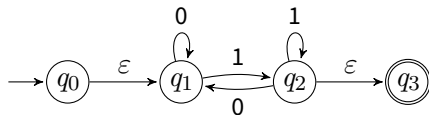
Eliminate  $q_1$ :



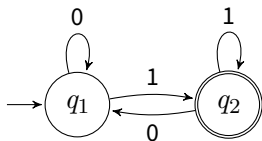
Eliminate  $q_2$ :



## Conversion example



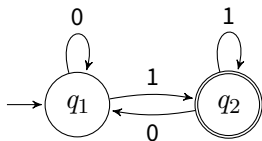
Check your answer!



All strings ending in 1  
 $(0 + 1)^*1$



## Check your answer!



All strings ending in 1  
 $(0 + 1)^*1$

$$0^*1(00^*1 + 1)^*$$

$$= 0^*1(0^*1)^*$$

Always ends in 1

Does every string ending in 1 have  
this form?

Yes