

# Lexical Analysis

CMPT 379: Compilers




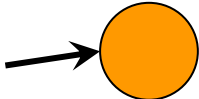
Instructor: Anoop Sarkar

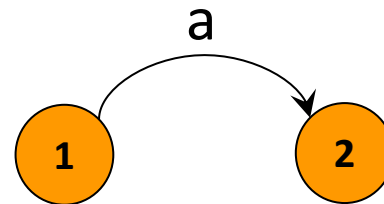
[anoopsarkar.github.io/compilers-class](https://anoopsarkar.github.io/compilers-class)

# Regular Expressions

- To describe all lexemes that form a token as a *pattern*
- Need decision procedure:  $s \in L(R)$  whether the given sequence of characters belongs to  $L(R)$  ?
  - Finite State Automata
  - Can be deterministic (DFA) or non-deterministic (NFA)

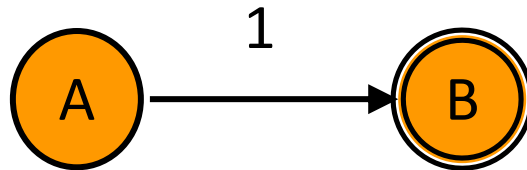
# Finite State Automata

- An alphabet  $\Sigma$  of input symbols
- A finite set of states  $S$  
  - One start state  $q_0$  
  - zero or more final (accepting) states  $F$  
- A transition function:
  - $\delta: S \times \Sigma \Rightarrow S$  
- Example:  $\delta(1, a) = 2$



# FA: Example

- A finite automaton that accepts only '1'

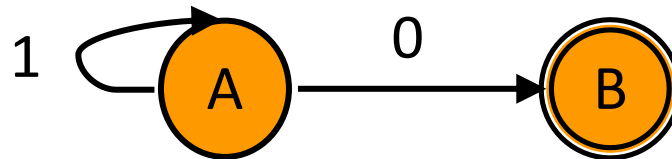


Language of a FA: set of accepted strings

state	input	
A	↑1	Accept
B	1↑	
A	↑0	Reject
A	↑1 0	Reject
B	1↑0	

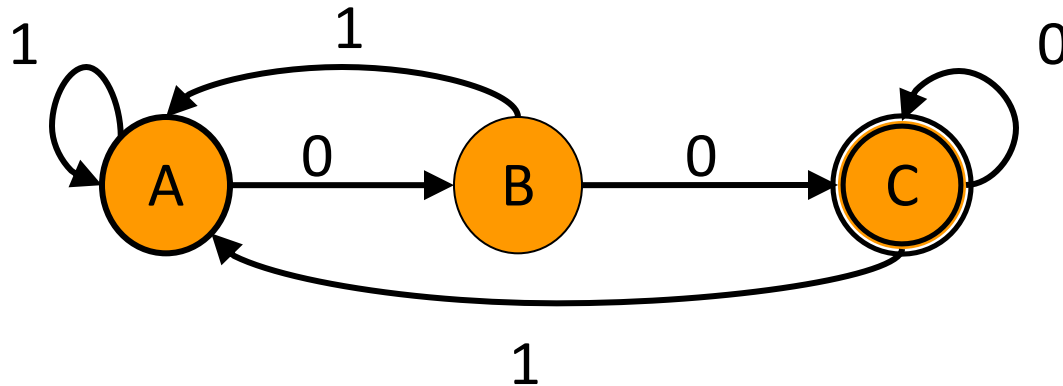
# FA: Example

- A finite automaton accepting any number of 1's followed by a single 0



# FA: Example

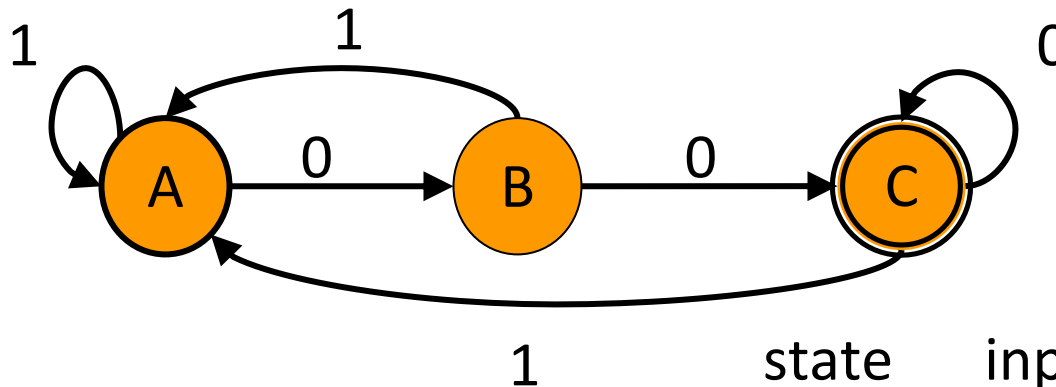
- What regular expression does this automaton accept?



A: start state  
C: final state

Answer:  $(0|1)^*00$

# FA simulation

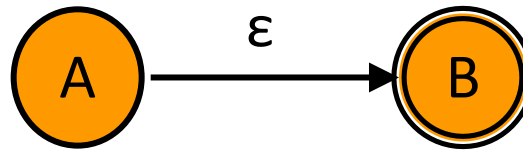


Input string: 00100

state	input
A	↑00100
B	00100 ↑
C	00100 ↑
A	00100 ↑
B	00100 ↑
C	00100 ↑
	Accept

# $\epsilon$ -move

- Another kind of transition:  $\epsilon$ -moves

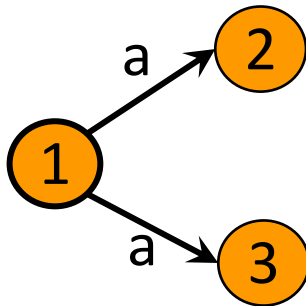


state	input
A	$x_1 x_2 x_3$ ↑
B	$x_1 x_2 x_3$ ↑



# Deterministic Finite Automata (DFA)

- One transition per input per state

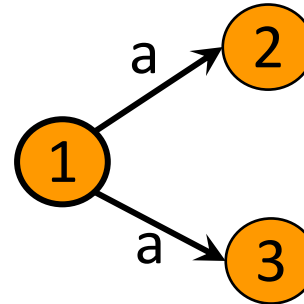


Invalid

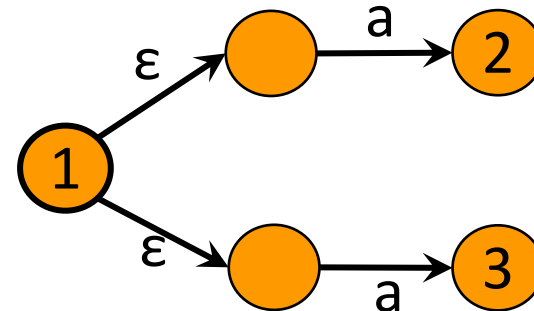
- No  $\epsilon$ -moves

# Nondeterministic Finite State Automata (NFA)

- Can have multiple transitions for one input in a given state

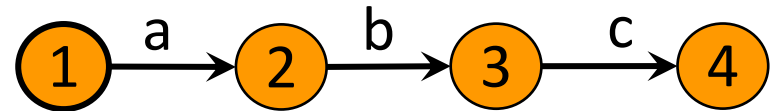


- Can have  $\epsilon$ -moves

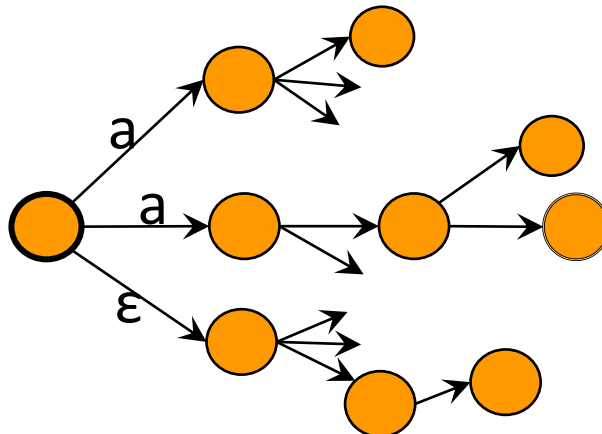


# Nondeterministic Finite State Automata (NFA)

- A DFA takes only one path through the state graph (per input)

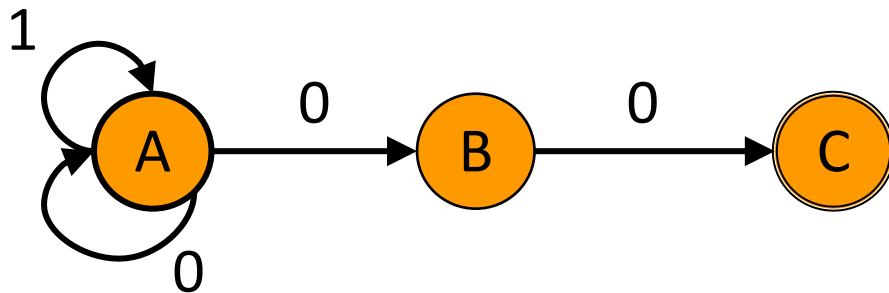


- NFA can choose!
  - An NFA accepts if some choices lead to a final state



# Nondeterministic Finite State Automata (NFA)

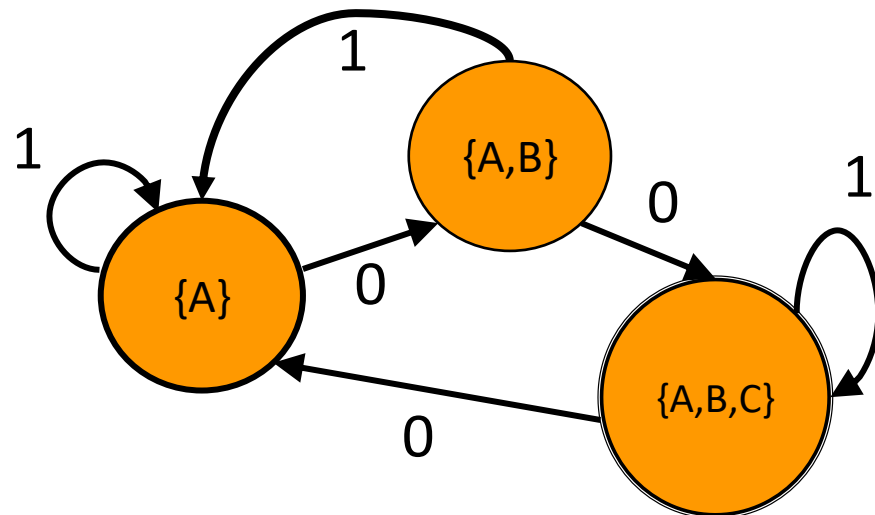
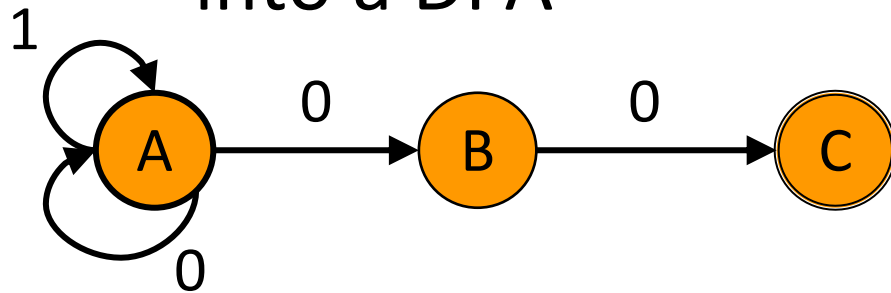
- An NFA can get into multiple states



state	input
{A}	↑100
{A}	↑100
{A,B}	↑100
{A,B,C}	↑100

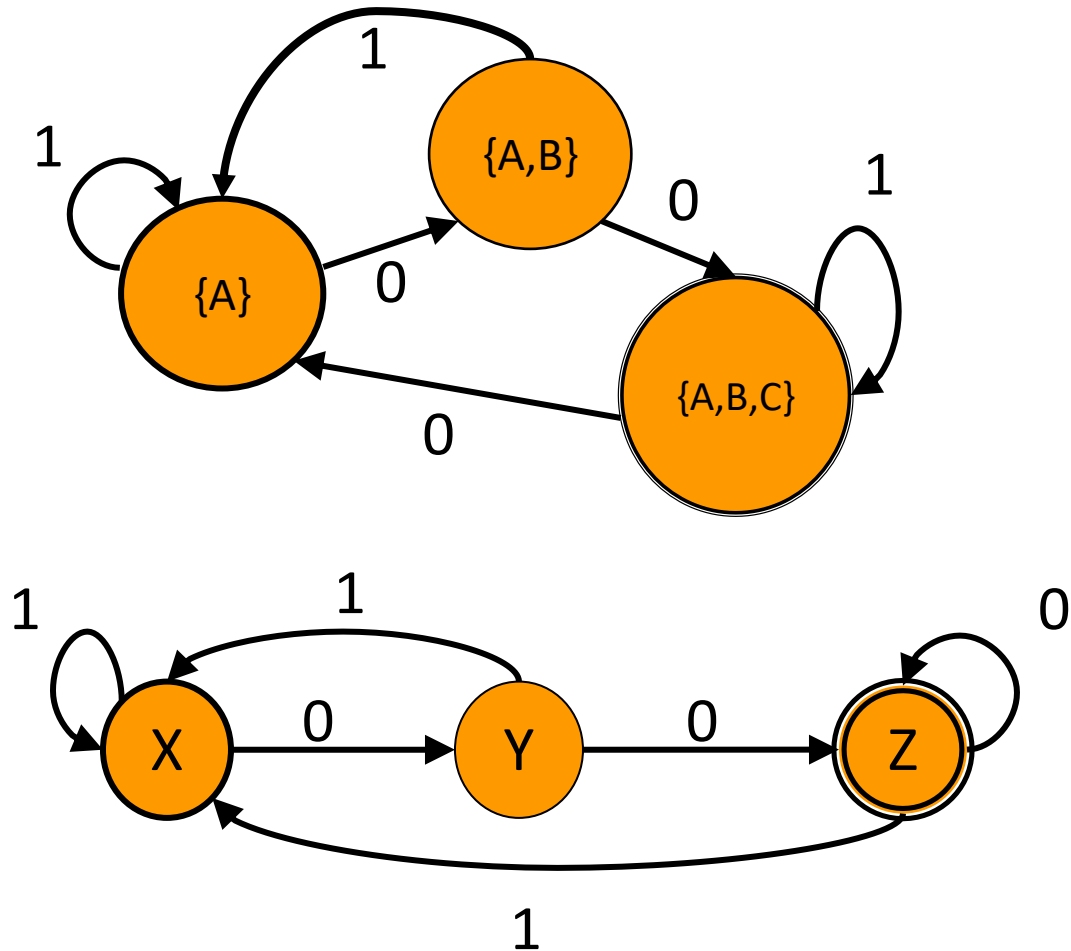
# Nondeterministic to Deterministic

- The Subset Construction converts an NFA into a DFA



DFA state, $\Sigma$	DFA state
$\{A\}, 0$	$\{A, B\}$
$\{A\}, 1$	$\{A\}$
$\{A, B\}, 0$	$\{A, B, C\}$
$\{A, B\}, 1$	$\{A\}$
$\{A, B, C\}, 0$	$\{A, B, C\}$
$\{A, B, C\}, 1$	$\{A\}$

# Nondeterministic to Deterministic



# NFAs vs DFAs

- NFAs and DFAs recognize the same set of languages
  - Regular expressions
- DFAs are faster to execute
  - There are no choices to consider
- DFAs are usually smaller than NFAs
- But in a worst case analysis, DFAs can be larger than NFAs
  - Exponentially larger