Nondeterministic Finite Automata

CSCI 3130 Formal Languages and Automata Theory

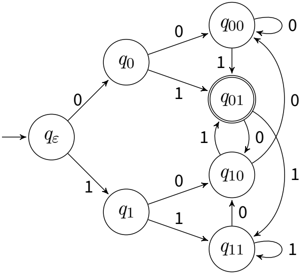
Siu On CHAN

Chinese University of Hong Kong

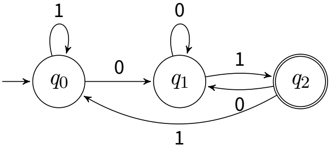
Fall 2015

Example from last lecture with a simpler solution

Construct a DFA over alphabet $\{0, 1\}$ that accepts all strings ending in 01



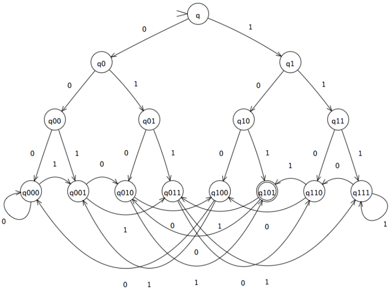
or



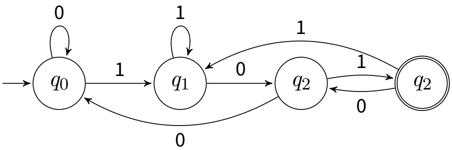
Three weeks later: DFA minimization

Another example from last lecture

Construct a DFA over alphabet $\{0, 1\}$ that accepts all strings ending in 101

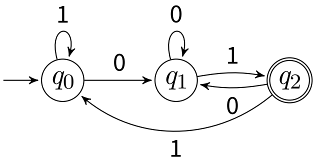


or

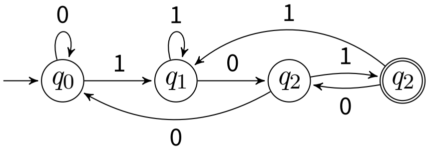


String matching DFAs

Ending in 01



Ending in 101



Fast string matching algorithms to turn a pattern into a string matching DFA and execute the DFA:

Boyer–Moore (BM) and Knuth–Morris–Pratt (KMP)

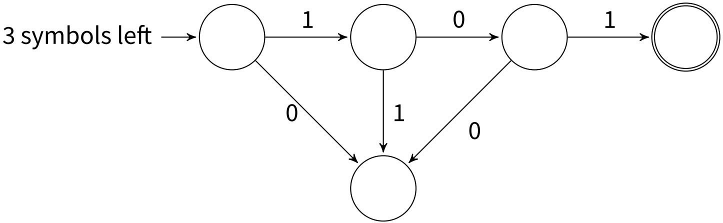
(won't cover in class)

Nondeterminism

Even easier with guesses

Suppose we could guess when the input string has only 3 symbols left

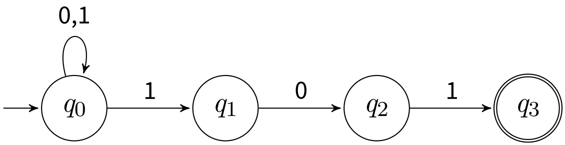
Accept strings ending in 101:



This is not a DFA!

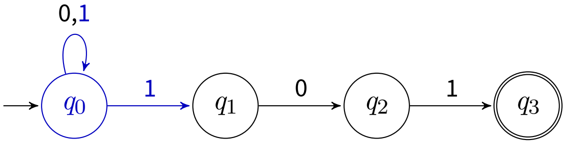
Nondeterministic finite automata

A machine that allows us to make guesses



Each state can have zero, one, or more outgoing transitions labeled by the same symbol

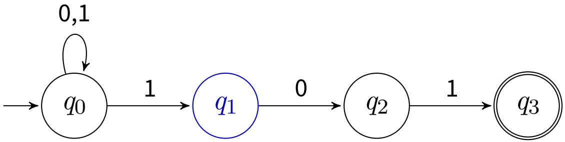
Choosing where to go



State q_0 has two transitions labeled 1

Upon reading 1, we have the choice of staying at q_0 or moving to q_1

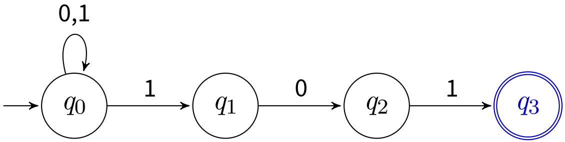
Ability to choose



State q_1 has no transition labeled 1

Upon reading 1 at q_1, die; upon reading 0, continue to q_2

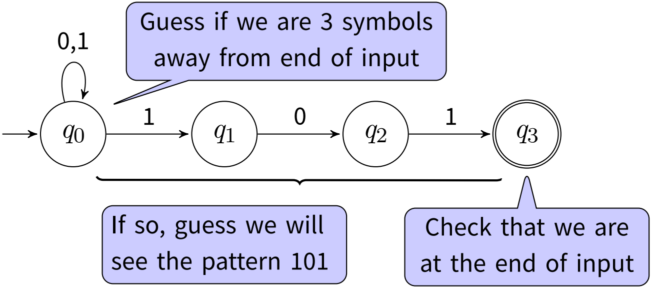
Ability to choose



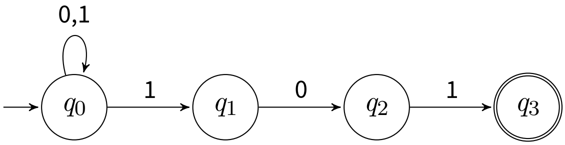
State q_1 has no transition going out

Upon reading 0 or 1 at q_3, die

Meaning of NFA



How to run an NFA



input: 01101

The NFA can have several active states at the same time

NFA accepts if at the end, one of its active states is accepting

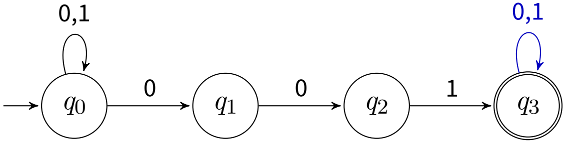
Example

Construct an NFA over alphabet $\{0, 1\}$ that accepts all strings containing the pattern 001 somewhere

11001010, 001001, 111001	should be accepted
ε, 000, 010101	should not

Example

Construct an NFA over alphabet $\{0, 1\}$ that accepts all strings containing the pattern 001 somewhere



Definition

A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to$ subsets of Q is a transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of accepting states

Differences from DFA:

- transition function δ can go into several states
- allows ε-transitions

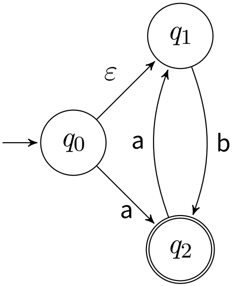
Language of an NFA

The NFA accepts string x if there is some path that, starting from q_0, ends at an accepting state as x is read from left to right

The language of an NFA is the set of all strings accepted by the NFA

ε-transitions

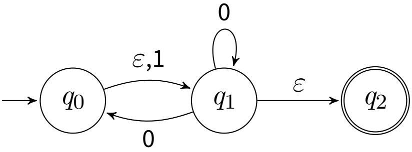
ε-transitions can be taken for free:



accepts
a, b, aab, bab, aabab, …

rejects
ε, aa, ba, bb, …

Example



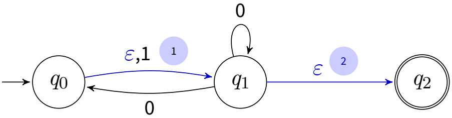
alphabet $\Sigma = \{0, 1\}$
states $Q = \{q_0, q_1, q_2\}$
initial state q_0
accepting states $F = \{q_2\}$

table of transition function δ

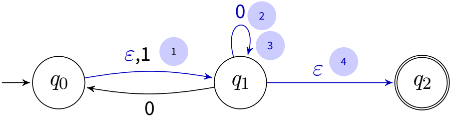
		inputs		
		0	1	ε
states	q_0	$\emptyset$	$\{q_1\}$	$\{q_1\}$
	q_1	$\{q_0, q_1\}$	$\emptyset$	$\{q_2\}$
	q_2	$\emptyset$	$\emptyset$	$\emptyset$

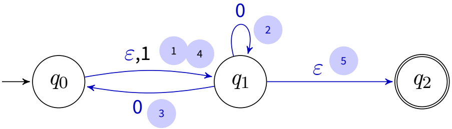
Running NFA

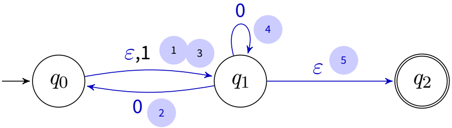
ε



00

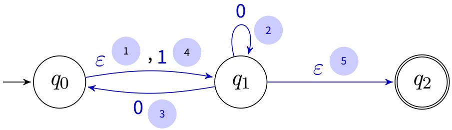




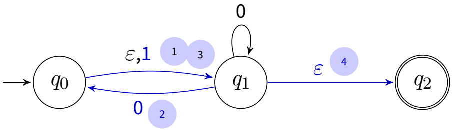


Running NFA

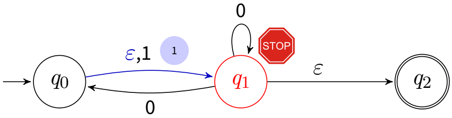
001

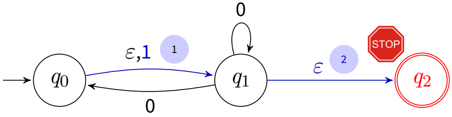


101

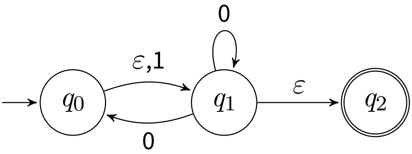


11





Language of this NFA



What is the language of this NFA?

Example of ε-transitions

Construct an NFA that accepts all strings with an even number of 0s or an odd number of 1s

Example of ε-transitions

Construct an NFA that accepts all strings with an even number of 0s or an odd number of 1s