Lecture 2

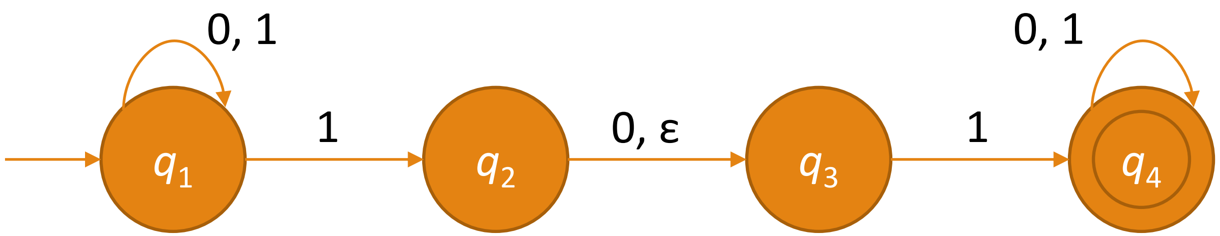
COT4210 DISCRETE STRUCTURES

DR. MATTHEW B. GERBER

5/24/2016

PORTIONS FROM SIPSER, *INTRODUCTION TO THE THEORY OF COMPUTATION*, 3RD ED., 2013

Nondeterminism



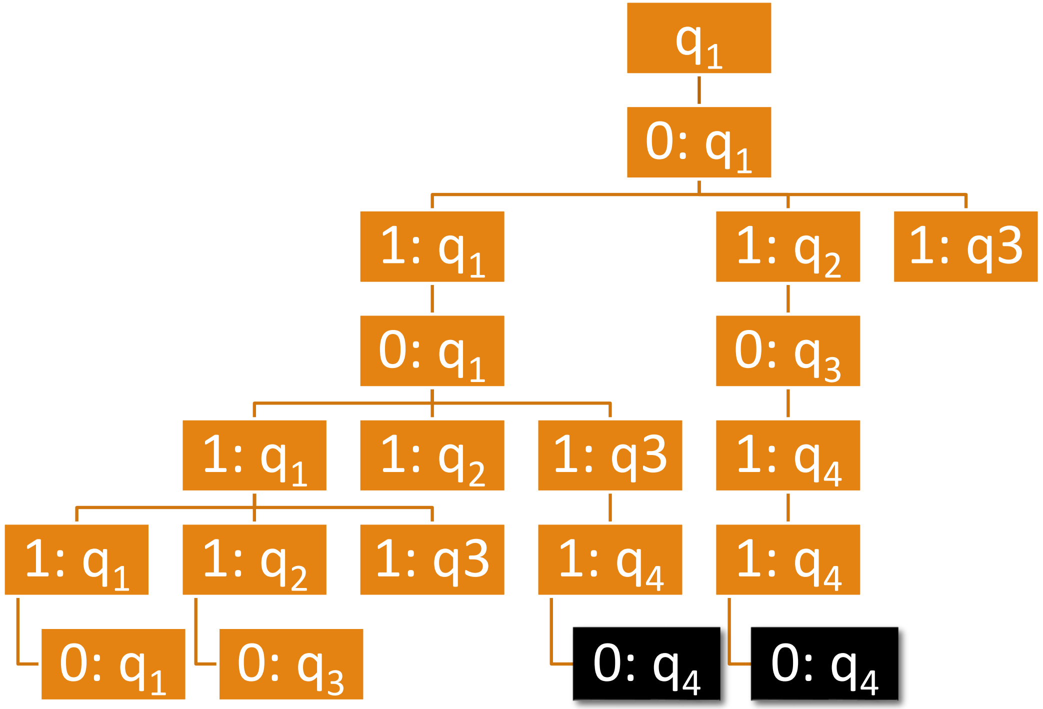
An NFA looks like a DFA, *except* an NFA:

- Can have more than one possible transition per state per input symbol
- Doesn't have to have a transition for every state for every input symbol
- Can transition on the empty string

It also *works* like a DFA, except acceptance is nondeterministic

- A DFA accepts if *the* path for the input string ends on an accept symbol
- An NFA accepts if *any* path for the input string ends on an accept symbol

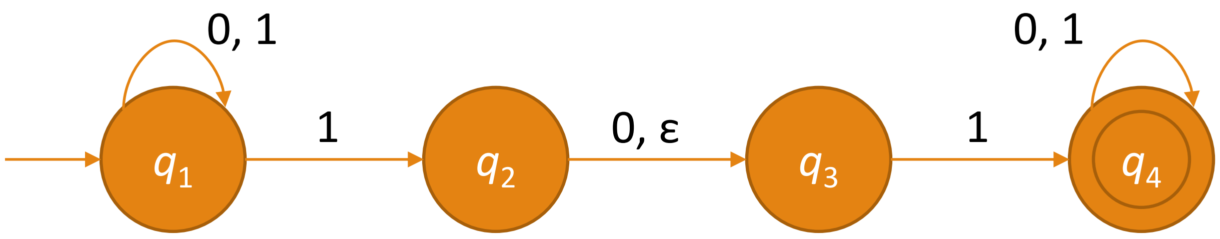
Computation in an NFA

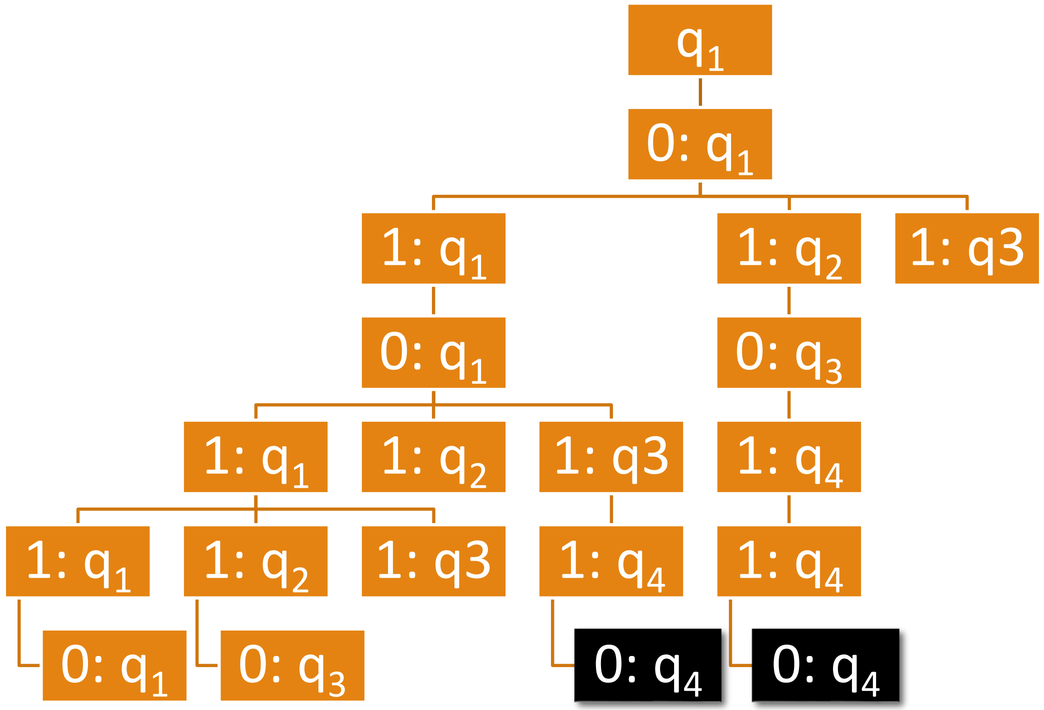


The easiest way to think of how an NFA works is to think of a *threaded DFA*

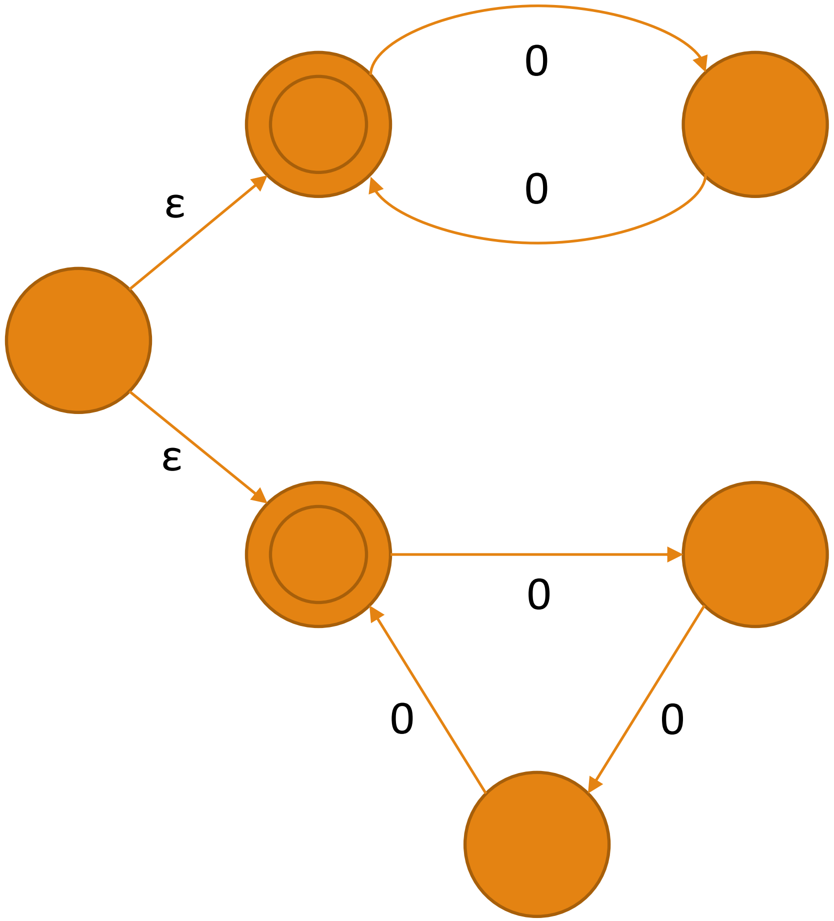
- Every time there is a choice of more than one path, the NFA splits off a copy of itself to follow each path
- The copies conceptually run in parallel
- A copy that reaches the end of input either accepts or rejects normally
- A copy that reaches a symbol it cannot transition on stops and rejects
- The NFA itself accepts if *any* copy accepts

Input 010110 on our NFA



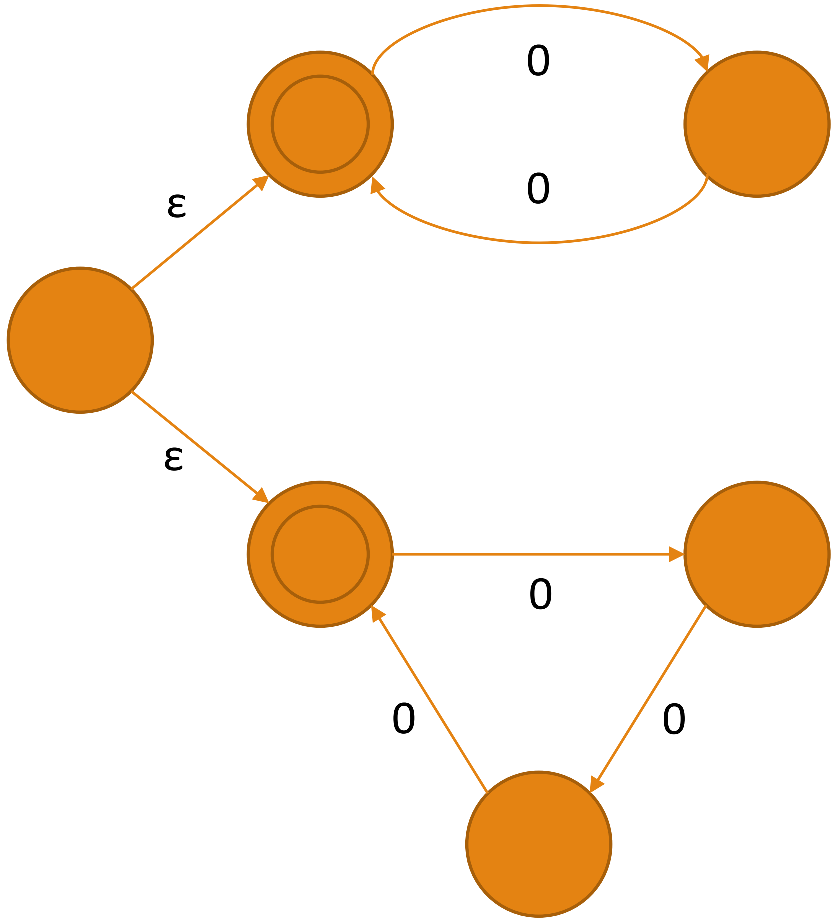


NFA Example



What does this machine do?

NFA Example



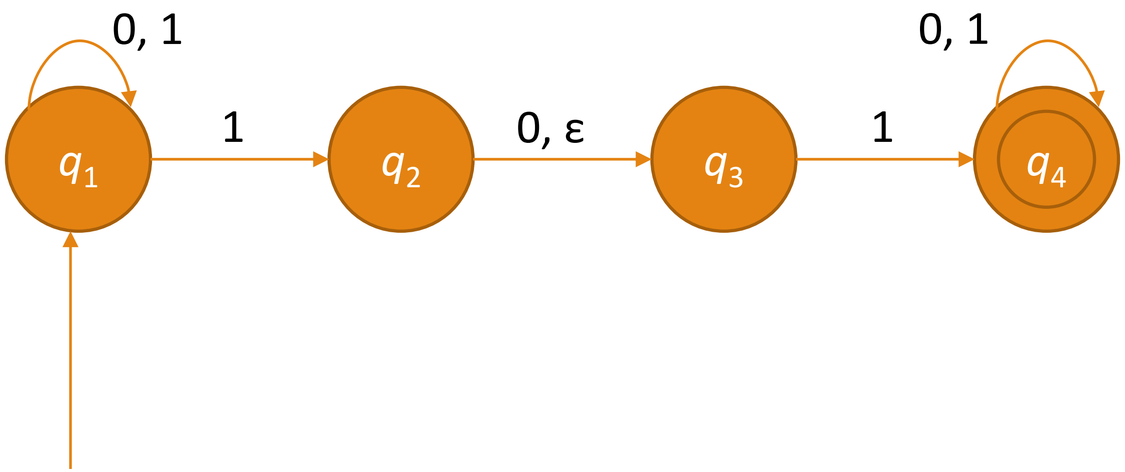
What does this machine do?

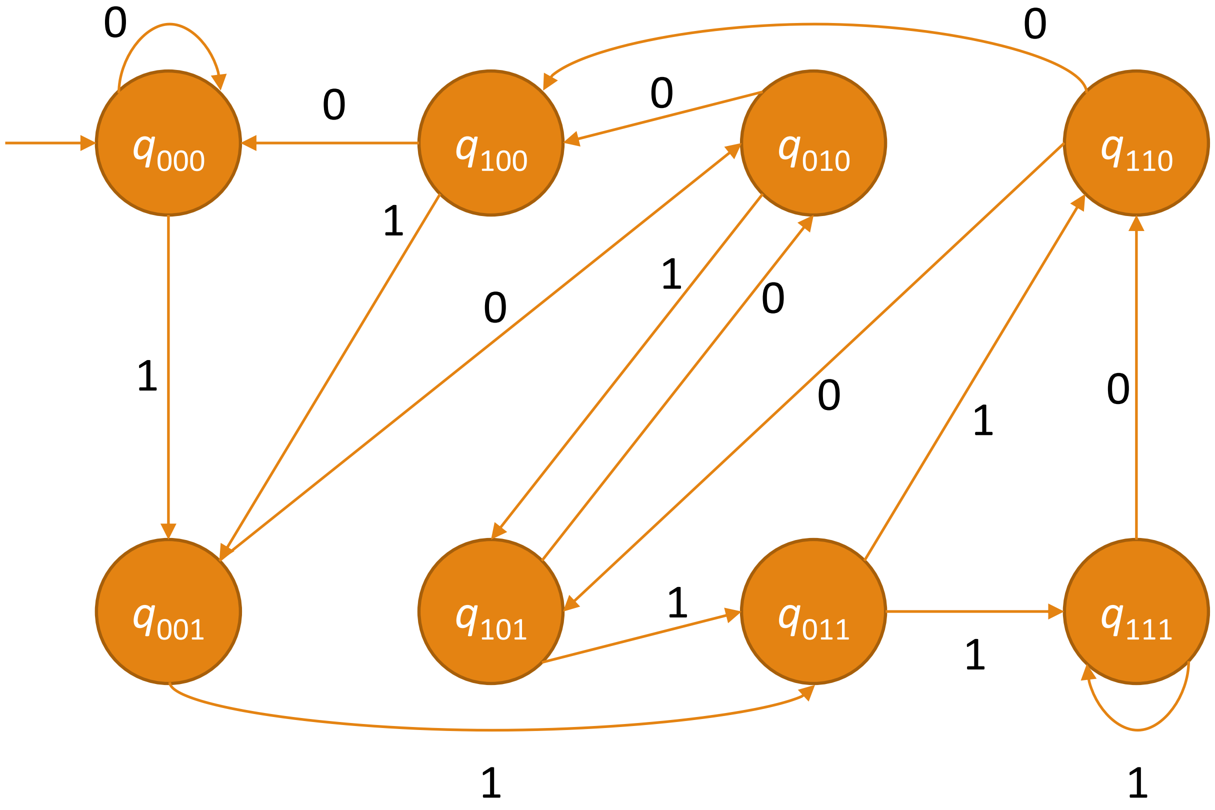
This machine accepts all strings consisting of a number of zeroes that's a multiple of either 2 or 3

- It's like our modulus machine from last lecture, except it accepts *x* mod 2 and *x* mod 3 at the same time

An NFA and its DFA

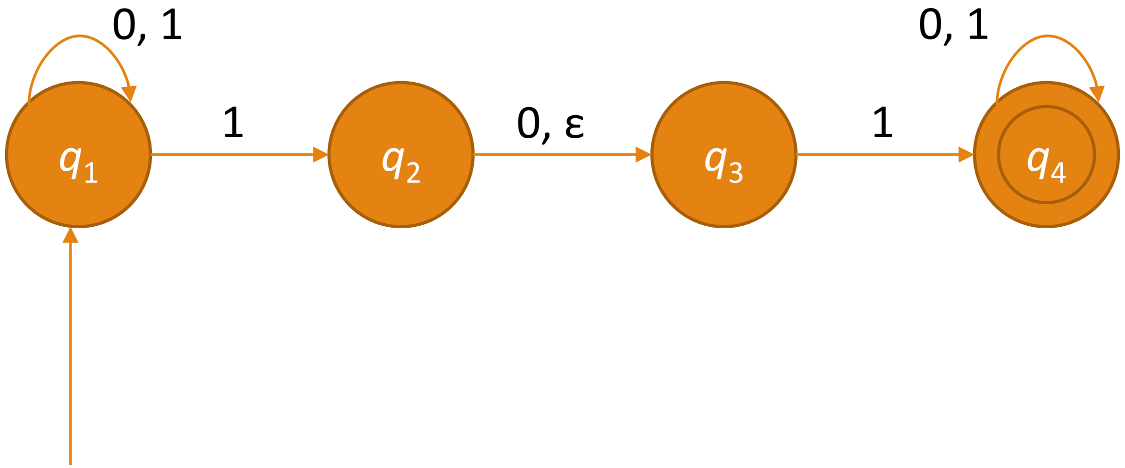
(Yes, this is, in fact, the best we can do.)





An NFA and its DFA

(Yes, this is, in fact, the best we can do.)



No, you're not missing some kind of subtle elegance.

This DFA is a hideous monster.

That's why we have NFAs.

Definition:
Nondeterministic Finite Automaton

A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that consists of:

◦ Q	A finite set of *states*
◦ Σ	An *alphabet*
◦ $\delta: Q \times \Sigma \to P(Q)$	A *transition function*
◦ $q_0 \in Q$	A *start state*
◦ $F \subseteq Q$	A set of *accept* (or *final*) states

Equivalence

The capabilities of NFAs are a strict superset of the capabilities of DFAs, so every DFA can obviously be made into an NFA.

- The reverse is less obvious – but is nonetheless true
- …and *immensely important*, as we'll get to soon enough

Proof: NFA/DFA Equivalence

Prove that every NFA has an equivalent DFA.

(Board work: Theorem 1.39)

Example: NFA/DFA Equivalence

(Board work: Example 1.41)

Consequences 1

If every NFA has a DFA, then every language that can be recognized with an NFA can be recognized with a DFA.

But that means…

Consequences 1

If every NFA has a DFA, then every language that can be recognized with an NFA can be recognized with a DFA.

But that means…

Corollary to NFA/DFA Equivalence: *A language is regular if and only if it can be recognized by an NFA.*

Consequences 2

We ended last session by taking about 20 minutes to prove that the class of regular languages was closed under union

- Let's do that again, a lot faster
- After that, we'll prove that it's closed under concatenation and star closure

(Board work: Theorems 1.45, 1.47, 1.49)

Next Time:
Regular Expressions