CS 4510: Automata and Complexity **Spring 2019**

Lecture 1: Models of Computation

August 19, 2019

Lecturer: Santosh Vempala *Scribe: Aditi Laddha*

1.1 Introduction

What is computation? A sequence of well defined state changes. Some examples of computation models: dominoes, motion of planets and celestial bodies, assembly lines, etc.

Consider the following problems, where we are given input in a specified format and are asked to compute the output.

1. **input**: a bit string of finite length
   output: YES iff number of 1's in the string is even

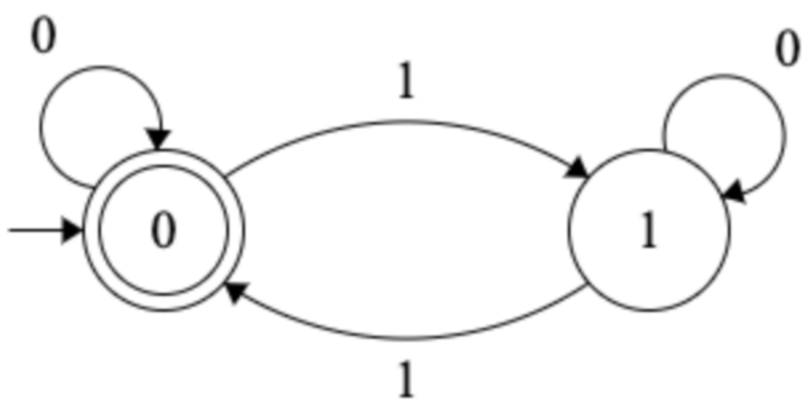

Figure 1.1: A DFA which accepts bit strings with even number of 1's

   If a string x stops at state i, then the number of ones in $x = i \mod 2$.

2. **input**: a non-negative decimal integer
   output: YES iff the integer is divisible by 3

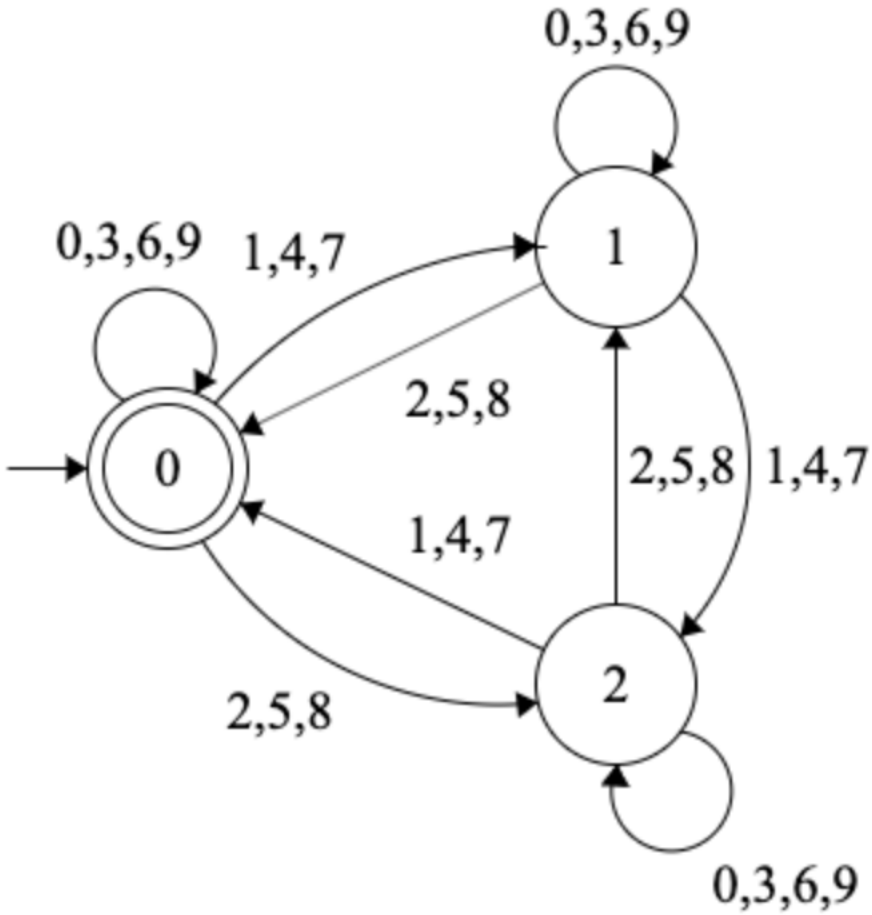

Figure 1.2: A DFA which accepts non-negative integral multiples of 3

An integer x is divisible by 3 if and only if the sum of its digits is divisible by 3. Here, if x stops at state i, then the sum of digits of $x = i \mod 3$.

1.2 Deterministic Finite Automata

A DFA is defined as a tuple $(Q, \Sigma, \sigma, F, q_0)$

- Q: A set of states of finite size
- Σ: input alphabet of finite size
- σ: $Q \times \Sigma \to Q$, transition function
- $F \subseteq Q$: set of accepting states
- q_0: the initial state

A DFA starts at state q_0. If the DFA has a transition $(q, a) \to q'$, it means that when the DFA reads the alphabet a in state q, the state transitions to q'. If a string causes the machine to stop in one of the accepting state, then we say that the automaton accept the string.

Can DFAs compute everything a computer can? Turns out the answer is no as we will see later lectures. We describe a more powerful computation model called a Turing machine.

1.3 Turing Machines

A Turing machine is defined as a tuple $(Q, \Gamma, \sigma, F, q_0)$

- Q: A set of states of finite size
- Γ: tape alphabet, $\Sigma \cup \{\_\}$
- $\sigma : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, transition function
- $F \subseteq Q$: set of accepting states
- q_0: the initial state

In addition to a state space, a Turing Machine also has a tape divided into cells where each cell contains a symbol from Γ and a head which marks the cell of the tape currently being read. The head can also write on a tape cell and move he tape left and right by one cell.

So, if the Turing machine has a transition $(q, a) \to (q', a', L)$, it means that when the tape head reads the alphabet a in state q, it rewrite a' in the current cell, moves to the cell left to the current one and changes the state of the machine to q'.

Turing machines are more powerful than DFAs and anything a real computer can compute, a Turing machine can also compute.