FLAT

1. Finite Automata and Regular Langauges

September 28, 2019

COURSE INFORMATION

ESSENTIALS

- Formal Language & Automata Theory (FLAT)
- Course Website - atu-se.github.io/courses/flat
- Primary Textbook - Introduction to the Theory of Computation, 3rd Edition (Sipser)]
- Lectures built around UC San Diego Lectures

MEETING TIMES

HOW TO SUCCEED

- Attend all lectures
- Take notes (not all materials will be distributed as slides)
- Ready and study your textbook
- Do all assignments
- Ask questions
- Work many practice problems

ACADEMIC INTEGRITY

- Read the academic integrity policy in the syllabus
- You are encouraged to discuss the topics among yourselves.
- Unless otherwise noted, your work should be your own and you should not share your work with others
- Copying or cheating on homework, exams, etc. may result in failing grades

FINITE AUTOMATA

INTRODUCTION

Computer Science stems from two starting points:

- Mathematics – What can be be computed? (And what *cannot* be computed?)
- Electrical Engineering – How can we build computers?

This course focuses on the first question

INTRODUCTION

- Computability Theory deals with the mathematical basis for Computer Science, yet it has some interesting practical ramifications that I will try to point out sometimes.
- The questions we will try to answer in this course are:
  - What can be computed?
  - What cannot be computed
  - Where is the line between the two?”

PRACTICAL APPLICATION

- Some areas of practical application include:
  - compiler design
  - natural language processing
  - static code analysis
  - IDE development (syntax highlighting)
  - cryptography
  - genetic research

COMPUTATIONAL MODELS

- A Computational Model is a mathematical object (defined on paper) that enables us to reason about computation and to study the properties and limitations of computing.
- We will deal with three principal computational models in increasing order of Computational Power.

COMPUTATIONAL MODELS

- We will deal with three principal models of computations:
  1. Finite Automaton (FA) – recognizes Regular Languages
  2. Push-down Automaton – recognizes Context Free Languages
  3. Turing Machine (TM) – recognizes Computable Languages

ALAN TURING - A SHORT DETOUR

Dr. Alan Turing is one of the founders of Computer Science (he was an English Mathematician)

1. “Invented” Turing machines.
2. “Invented” the Turing Test.
3. Broke the German submarine transmission coding machine “Enigma”.
4. The movie Imitation Game is loosely based on his life.

FINITE AUTOMATON EXAMPLE

WASHING MACHINE

- The control of a washing machine is a very simple example of a finite automaton.
- The most simple washing machine accepts quarters and operation does not start until at least 3 quarters were inserted.
- Credit: Vadim Lyubasehvsky

COINS

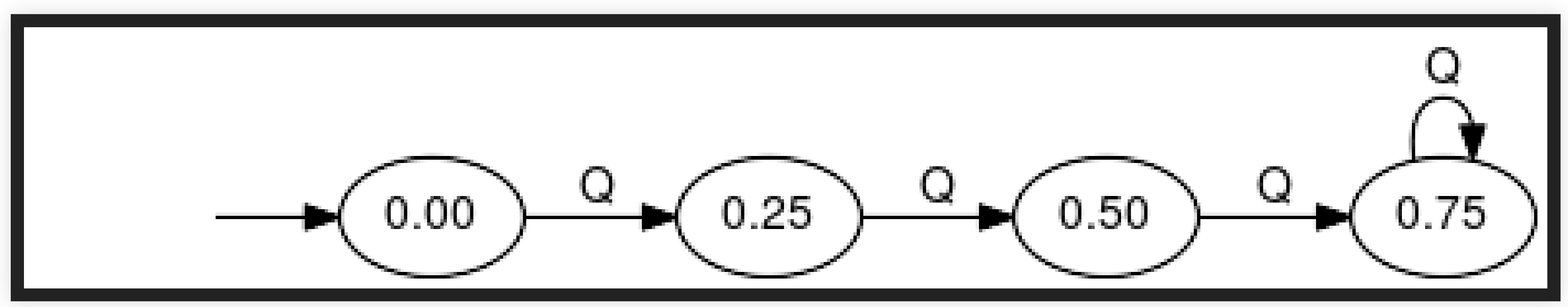
Washing machines take coins. Our machine costs $0.75 to operate. We will use this notation:

- Q = Quarter (0.25)
- H = Half-dollar (0.50)
- D = Dollar (1)

WASHING MACHINE 1

- Accepts Three Quarters
- Put in three (or more quarters) – it begins operation
- Put in less – it does nothing

WASHING MACHINE 1

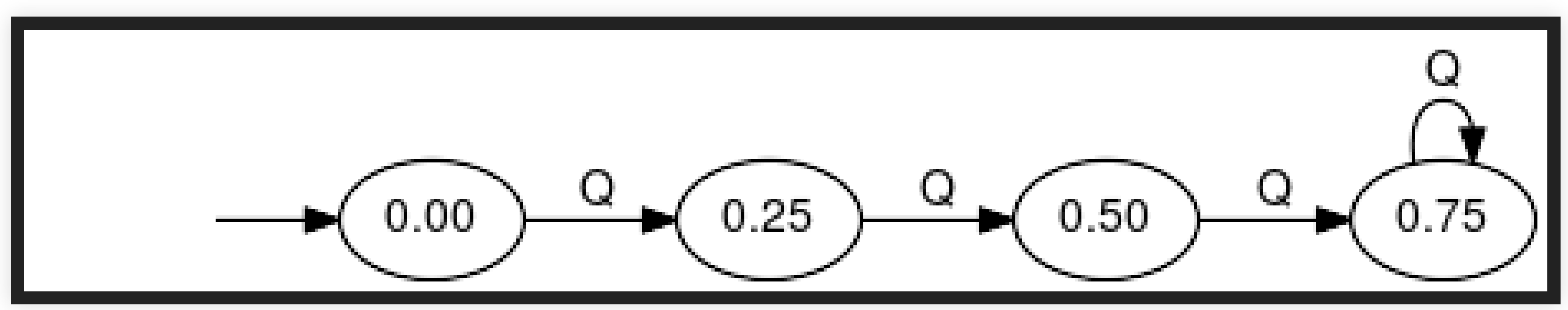


WASHING MACHINE 2

- Now imagine we have a more advanced machine which can accept both quarters (Q) and half-dollars (H).
- How does the automata change?

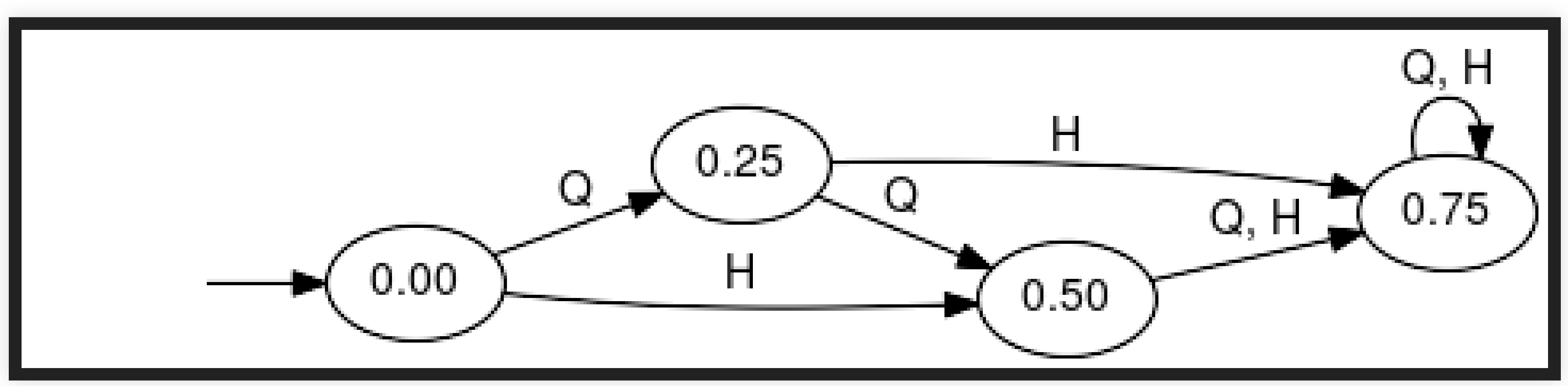
WASHING MACHINE 2

Modify Washing Machine 1:



WASHING MACHINE 2

Result:

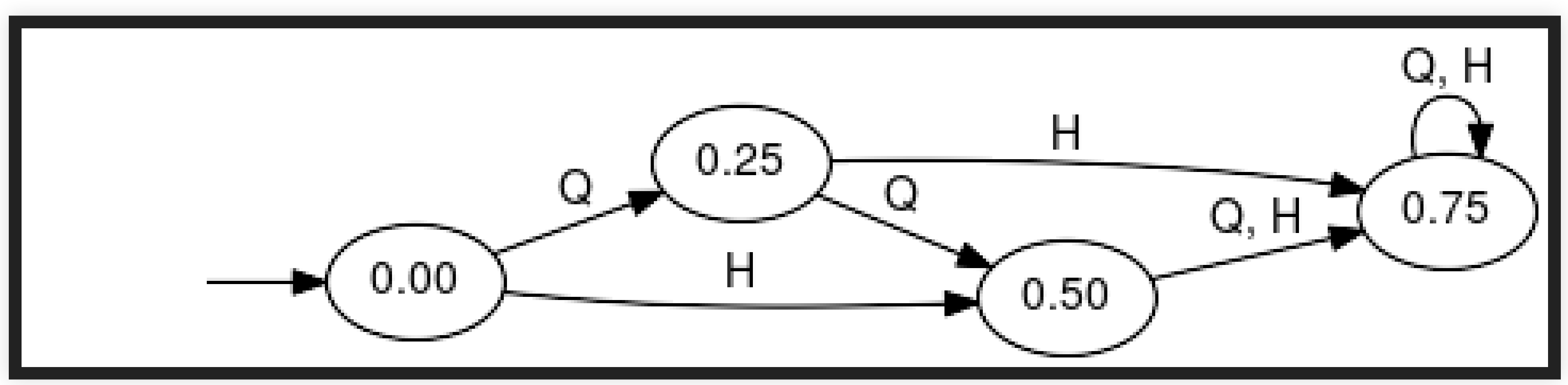


WASHING MACHINE 3

- Now imagine we have a more advanced machine which can accept quarters (Q), and half-dollars (H), and dollars (D).
- How does the automata change?

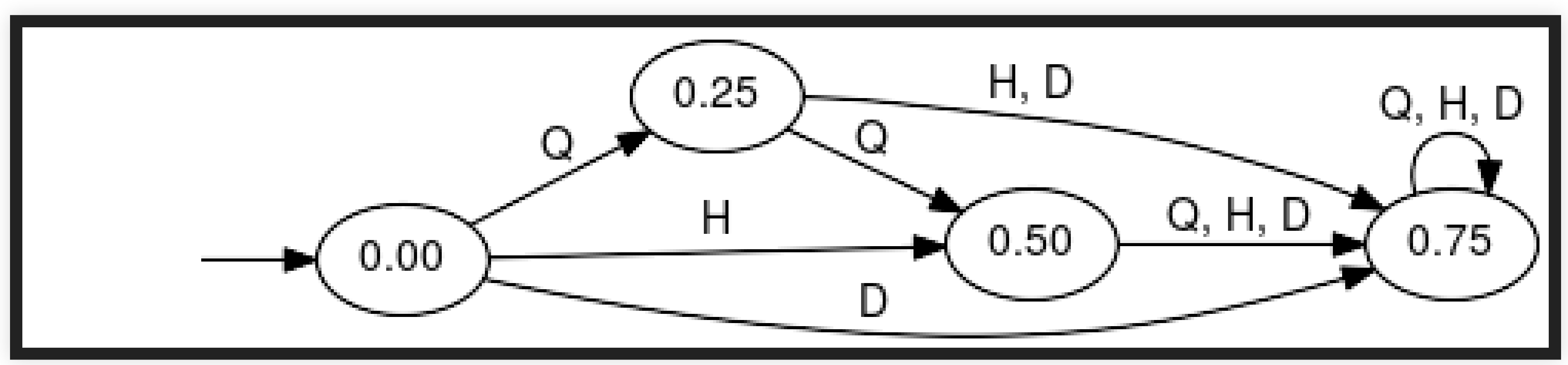
WASHING MACHINE 3

Modify Washing Machine 2:



WASHING MACHINE 3

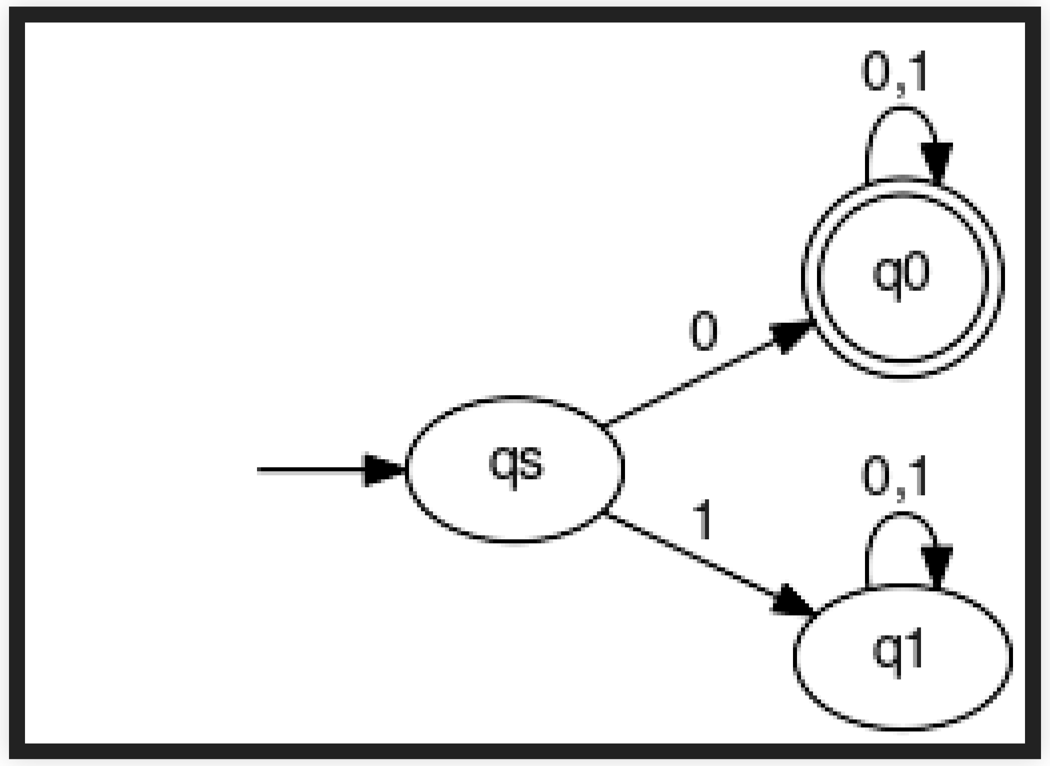
Result:



FA EXAMPLE

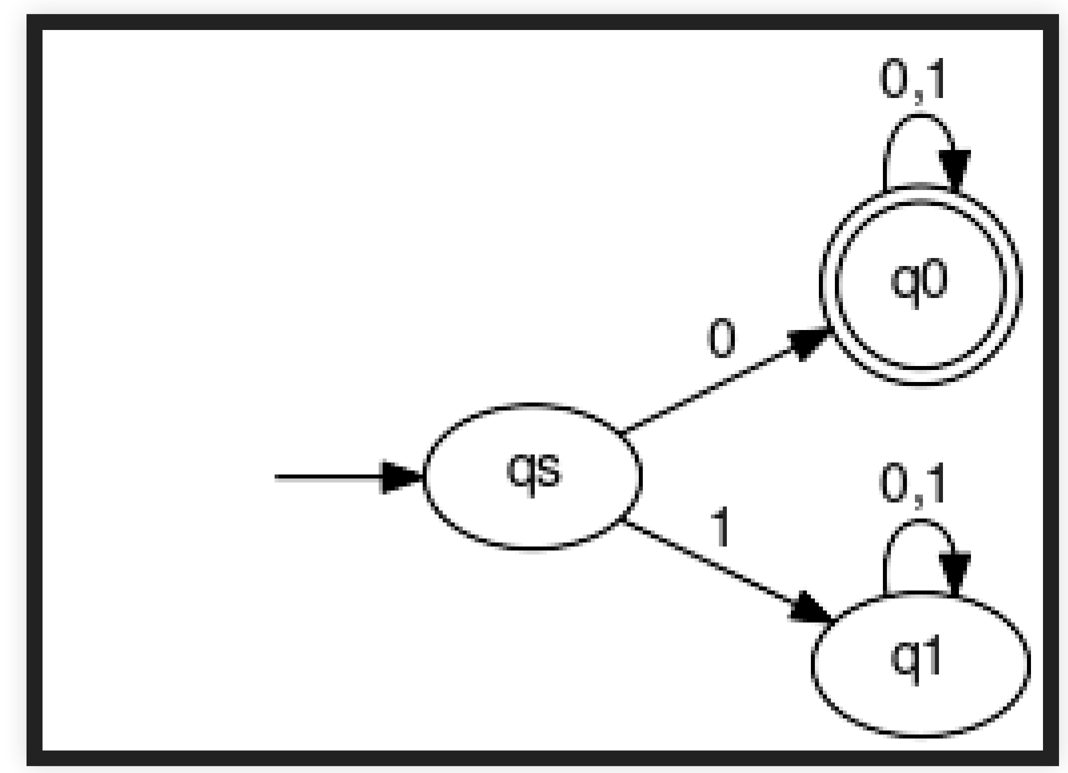
M_1:

- States: $\{q_s, q_0, q_1\}$
- Initial State: q_s
- Final/Final States: $\{q_0\}$



FA EXAMPLE

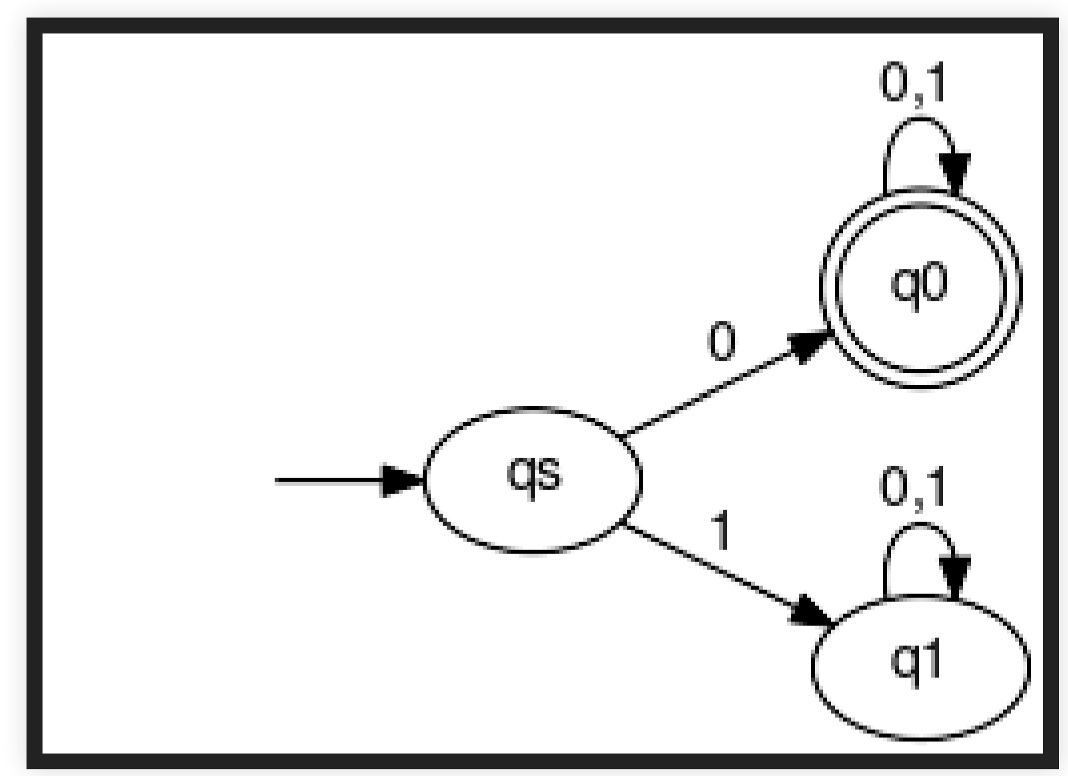
- M_1 Transition Function:
  - $\delta(q_s, 0) = q_0$
  - $\delta(q_s, 1) = q_1$
  - $\delta(q_0, 0) = \delta(q_0, 1) = q_0$
  - $\delta(q_1, 0) = \delta(q_1, 1) = q_1$



FINITE AUTOMATON EXAMPLE

M_1:

- Alphabet: $\{0, 1\}$
- Accepted Words: $\{0, 00, 01, 000, 001, \ldots\}$



FA - FORMAL DEFINITION

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

1. Q is a finite set called the *states*.
2. Σ is a finite set called the *alphabet*.
3. $\delta : Q \times \Sigma \rightarrow Q$ is the *transition function*.
4. $q_0 \in Q$ is the *start state*, and
5. $F \subseteq Q$,is the set of *accept states*.

FA - FORMAL DEFINITION

- Don’t get lost in the notation.
- More important than the notation is to understand the concepts
- Notation allows us to communicate clearly
  - Σ = “Sigma”
  - δ = “Delta”

OBSERVATIONS

1. Each state has *a single transition* for each symbol in the alphabet
2. Every FA has a computation for *every finite string* over the alphabet

EXAMPLES

M_2 accepts all words (strings) ending with 1 over the alphabet $\{0, 1\}$.

The language recognized by M_2 called $L(M_2)$ satisfies:

$$L(M_2) = \{w | w \textit{ ends with a } 1\}$$

HOW TO DO IT

1. Find some simple examples (short accepted and rejected words)
2. Think what each state should “remember” or represent
3. Draw the states with a meaningful name
4. Draw transitions that preserve the states’ “memory”
5. Validate or correct
6. Write a correctness argument

EXERCISES

1. M_3 accepts all words (strings) beginning with 1.
   $L(M_3) = \{w | w \textit{ ends with } 1\}$
2. M_4 accepts all words ending with 0.
3. M_5 accepts all words not ending with 0, including ϵ
   - You can accept an ϵ by accepting the start state
   - M_5 is the *complement automaton* of M_4

EXERCISES

4. M_6 accepts all string *over alphabet* $\{a, b\}$ that *start and end* with the same *symbol*.
5. M_7 accepts words of the form 0^m1^n where m, n are integers and $m, n > 0$
6. M_8 accepts all words in $\{0, 1, 00, 01, 10\}$