Final Project Revisited;
Finite State Machines, part III
Turing Machines;

Computer Science 111
Boston University

Vahid Azadeh Ranjbar, Ph.D.

Final Project: Classifying a New Body of Text

Suppose we're just focused on the word frequencies:

William Shakespeare
WS: { "love": 50,
"spell": 8,
"thou": 42 }

J.K. Rowling
JKR: { "love": 25,
"spell": 275,
"potter": 700 }

New: { "love": 3, "thou": 1,
"potter": 2 }

How could we give a similarity score for this **new** dictionary against each one above?

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

WS: { "love": 50,
"spell": 8,
"thou": 42 }

score vs. WS

New: { "love": 3, "thou": 1,
"potter": 2 }

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

WS: { **"love": 50**,
"spell": 8,
"thou": 42 }

100 words in all

score vs. WS

$$\underbrace{\frac{50}{100}}_{\text{love}} \cdot \underbrace{\frac{50}{100}}_{\text{love}} \cdot \underbrace{\frac{50}{100}}_{\text{love}}$$

New: { **"love"**: **3**, "thou": 1,
"potter": 2 }

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

WS: { "love": 50,
"spell": 8,
"thou": 42 }

100 words in all

score vs. WS

$$\frac{50}{100} \cdot \frac{50}{100} \cdot \frac{50}{100} \cdot \frac{42}{100}$$

love love love thou

New: { "love": 3, **"thou": 1**,
"potter": 2 }

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

"potter" is not here.

WS: { "love": 50,
"spell": 8,
"thou": 42 }

100 words in all

score vs. WS

$$\frac{50}{100} \cdot \frac{50}{100} \cdot \frac{50}{100} \cdot \frac{42}{100} \cdot \frac{0}{100} \cdot \frac{0}{100}$$

love love love thou potter potter

New: { "love": 3, "thou": 1,
"potter": 2 }

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

"potter" is not here.

WS: { "love": 50,
"spell": 8,
"thou": 42 }

100 words in all

score vs. WS

$$\underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{thou}}{\frac{42}{100}} \cdot \underset{\text{potter}}{\frac{0}{100}} \cdot \underset{\text{potter}}{\frac{0}{100}}$$

score = **0**

New: { "love": 3, "thou": 1,
"potter": 2 }

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

"potter" is not here.

WS: { "love": 50,
"spell": 8,
"thou": 42 }

100 words in all

score vs. WS

$$\underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{thou}}{\frac{42}{100}} \cdot \underset{\text{potter}}{\frac{0.5}{100}} \cdot \underset{\text{potter}}{\frac{0.5}{100}}$$

score = 0.00000131

New: { "love": 3, "thou": 1,
"potter": 2 }

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

"potter" is not here.

WS: { "love": 50,
"spell": 8,
"thou": 42 }

100 words in all

"thou" is not here.

JKR: { "love": 25,
"spell": 275,
"potter": 700 }

1000 words in all

score vs. WS

$$\underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{thou}}{\frac{42}{100}} \cdot \underset{\text{potter}}{\frac{0.5}{100}} \cdot \underset{\text{potter}}{\frac{0.5}{100}}$$

score = 0.00000131

>

score vs. JKR

$$\underset{\text{love}}{\frac{25}{1000}} \cdot \underset{\text{love}}{\frac{25}{1000}} \cdot \underset{\text{love}}{\frac{25}{1000}} \cdot \underset{\text{thou}}{\frac{0.5}{1000}} \cdot \underset{\text{potter}}{\frac{700}{1000}} \cdot \underset{\text{potter}}{\frac{700}{1000}}$$

score ~= 0.00000000382

New: { "love": 3, "thou": 1,
"potter": 2 }

more likely to be WS!

problem: scores can become too small!

Naïve Bayes Scoring Algorithm

multiply each word's probability as if they were all independent

"potter" is not here.

WS: { "love": 50,
"spell": 8,
"thou": 42 }

100 words in all

"thou" is not here.

JKR: { "love": 25,
"spell": 275,
"potter": 700 }

1000 words in all

score vs. WS

$$\underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{love}}{\frac{50}{100}} \cdot \underset{\text{thou}}{\frac{42}{100}} \cdot \underset{\text{potter}}{\frac{0.5}{100}} \cdot \underset{\text{potter}}{\frac{0.5}{100}}$$

$$3\log\left(\frac{50}{100}\right) + 1\log\left(\frac{42}{100}\right) + 2\log\left(\frac{0.5}{100}\right)$$

score ~= -13.54

>

score vs. JKR

$$\underset{\text{love}}{\frac{25}{1000}} \cdot \underset{\text{love}}{\frac{25}{1000}} \cdot \underset{\text{love}}{\frac{25}{1000}} \cdot \underset{\text{thou}}{\frac{0.5}{1000}} \cdot \underset{\text{potter}}{\frac{700}{1000}} \cdot \underset{\text{potter}}{\frac{700}{1000}}$$

$$3\log\left(\frac{25}{1000}\right) + 1\log\left(\frac{0.5}{1000}\right) + 2\log\left(\frac{700}{1000}\right)$$

score ~= -19.38

New: { "love": 3, "thou": 1,
"potter": 2 }

more likely to be WS!

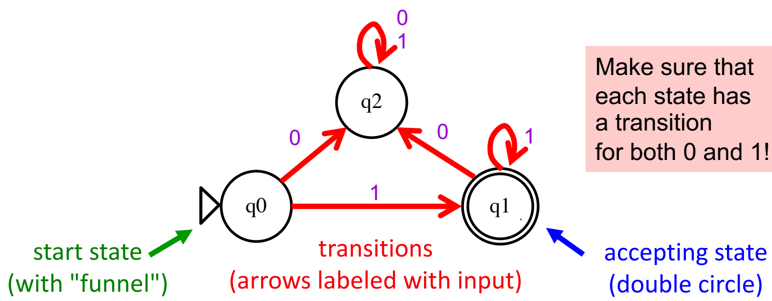
problem: scores can become too small!
solution: use logs!

Recall: Finite State Machine (FSM)

- An abstract model of computation
- Consists of:
  - one or more states

    exactly one of them is the start / initial state

    zero or more of them can be an accepting state
  - a set of possible input characters (we're using {0, 1})
  - transitions between states, based on the inputs



Make sure that each state has a transition for both 0 and 1!

More FSM Practice!

- Construct a FSM accepting bit strings in which:
  - the **first** bit is 0
  - the **last** bit is 1
- ***What are the classes of equivalent inputs?***

  empty string (q0)

  first bit is 1 (q1)

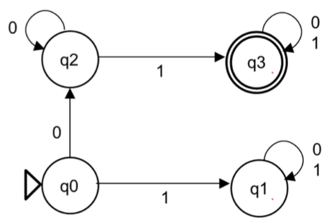
  first bit is 0, last bit is 0 (q2)

  first bit is 0, last bit is 1 (q3)

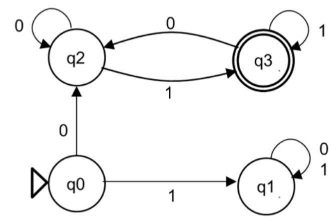
Which of these is the correct FSM?

- Construct a FSM accepting bit strings in which:
  - the **first** bit is 0
  - the **last** bit is 1

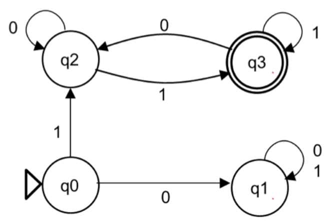
A.



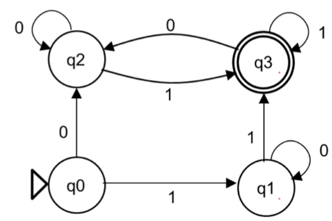
C.



B.



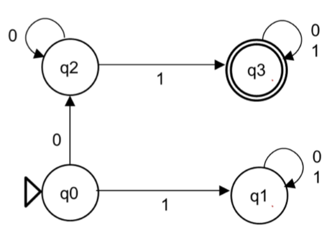
D.



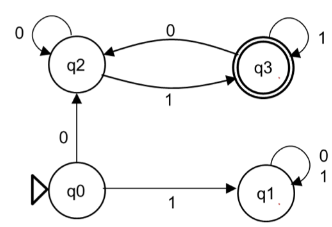
Which of these is the correct FSM?

- Construct a FSM accepting bit strings in which:
  - the **first** bit is 0
  - the **last** bit is 1

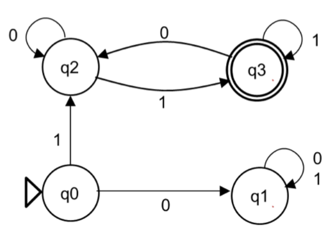
A.



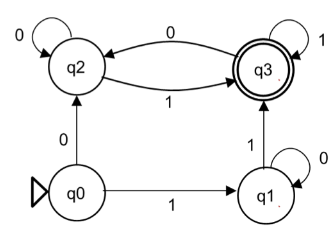
C.



B.



D.

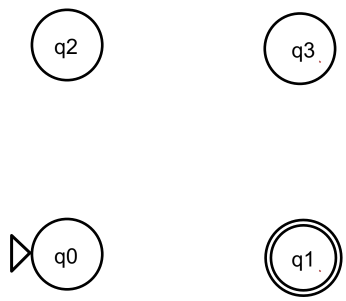


Even More Practice!

- Construct a FSM accepting bit strings in which:
  - the number of 1s is odd
  - the number of 0s is even

- Start by asking: ***What are the classes of equivalent inputs?***
  - 1s even, 0s even (q0)
  - 1s odd, 0s even (q1)
  - 1s even, 0s odd (q2)
  - 1s odd, 0s odd (q3)

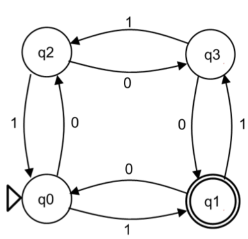
Now draw the FSM!



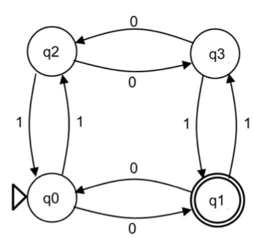
Which of these is the correct FSM?

- Construct a FSM accepting bit strings in which:
  - the number of 1s is odd
  - the number of 0s is even

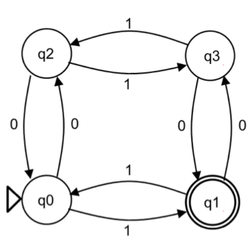
A.



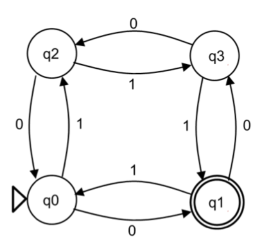
C.



B.



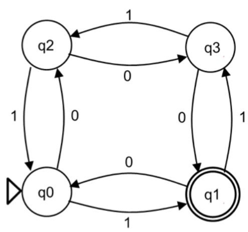
D.



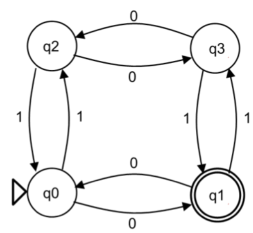
Which of these is the correct FSM?

- Construct a FSM accepting bit strings in which:
  - the number of 1s is odd
  - the number of 0s is even

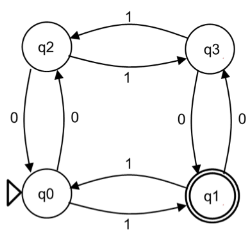
A.



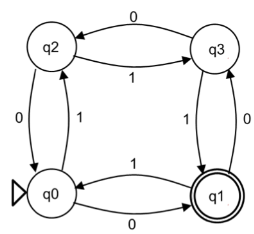
C.



B.



D.

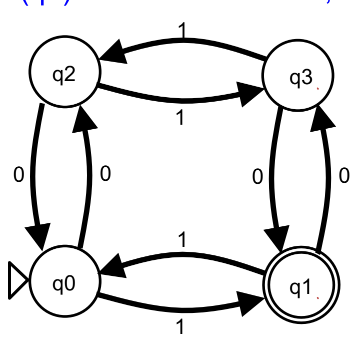


Even More Practice!

- Construct a FSM accepting bit strings in which:
  - the number of 1s is odd
  - the number of 0s is even

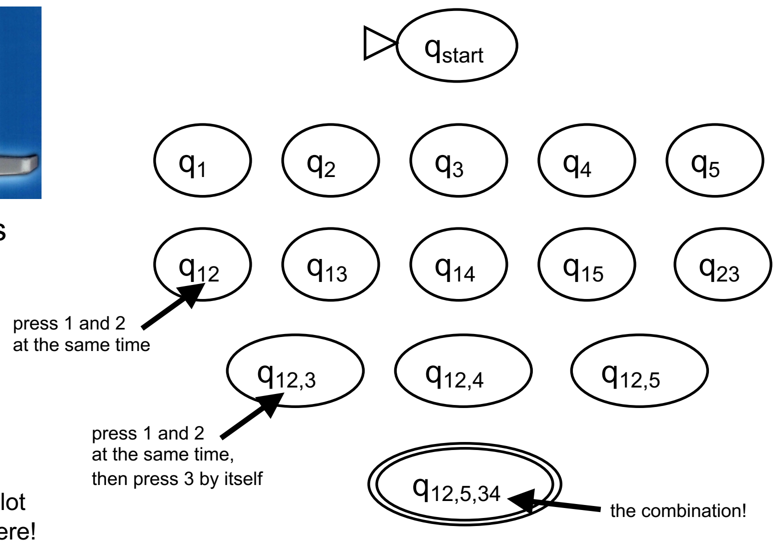
- Start by asking: ***What are the classes of equivalent inputs?***

  1s even, 0s even (q0)
  1s odd, 0s even (q1)
  1s even, 0s odd (q2)
  1s odd, 0s odd (q3)



FSMs are everywhere!

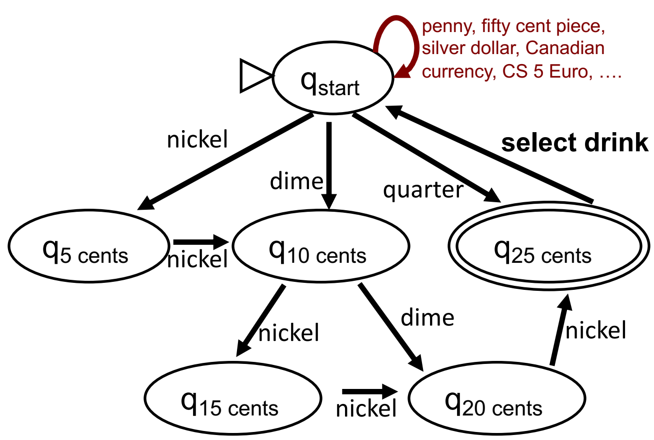
Locks



There's a lot missing here!

FSMs are everywhere!

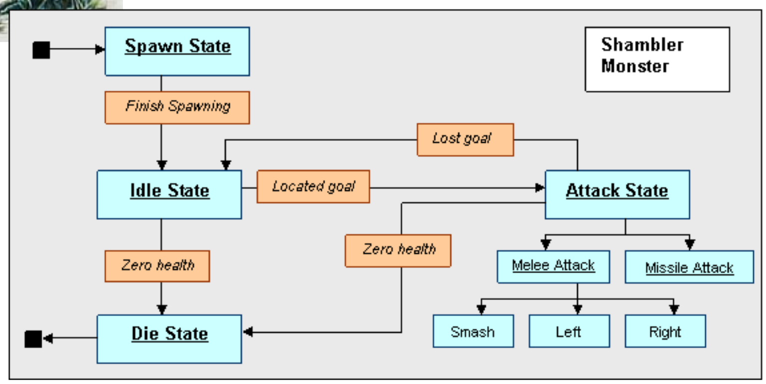
mechanical vending machine



(some transitions not shown)

FSMs in Game AI

The state machine controlling *Shambler* monsters in Quake...



FSMs in Robotics: Towel-Folding!

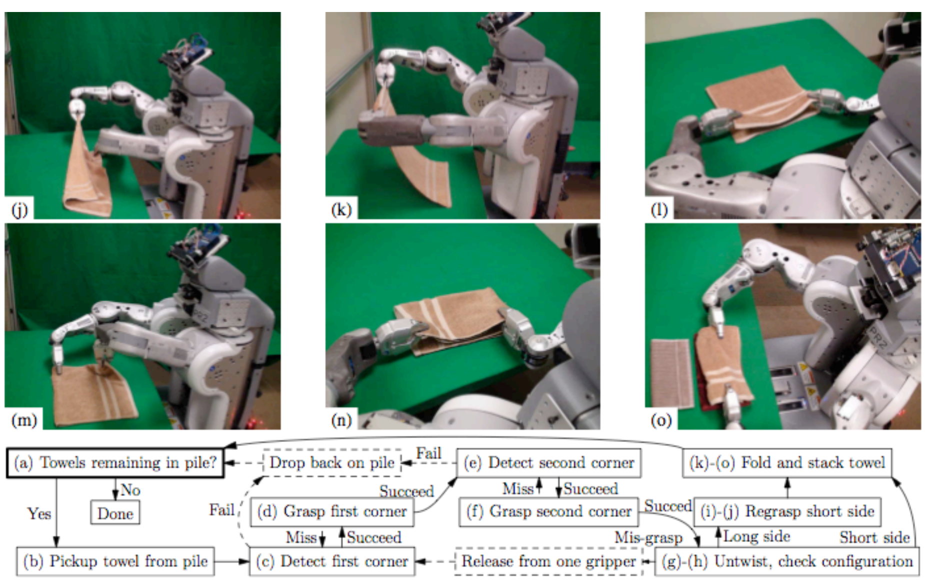


Fig. 2. The state machine model of the procedure: dashed lines indicate failure recovery cases. The images show an actual run.

An Autonomous Vehicle's FSM

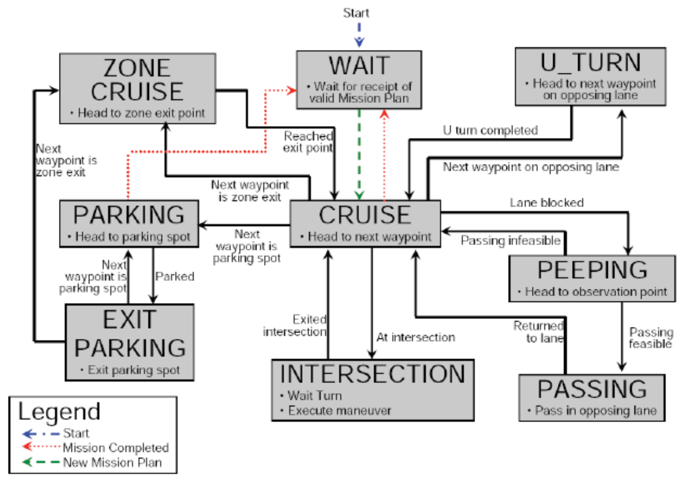


Fig. 9. Situational Interpreter State Transition Diagram. All modes are sub-modes of the system RUN mode (Fig 4(b)).

What About This Problem?

- Construct a FSM accepting bit strings that:
  - start with **some** number of 0s
  - followed by the **same** number of 1s
  - 01, 0011, 000111, 00001111, etc.

- ***What are the classes of equivalent inputs?***

  an infinite number of them!

  n **0**s, followed by (*n*+1) or more **1**s, and/or by an alternation between groups of **1**s and **0**s – **rejected; can't recover!**

  n **0**s, followed by *n* **1**s – **accepted!** (and any further input is bad!)

  n **0**s, followed by (*n*–1) **1**s – need *one* more **1** to accept

  n **0**s, followed by (*n*–2) **1**s – need *two* more **1**s to accept

  n **0**s, followed by (*n*–3) **1**s – need *three* more **1**s to accept

  …

- **Impossible to solve using a *finite* state machine!**

Limitations of FSMs

- Because they're finite, FSMs can only count finitely high!

Computable with FSMs

even/odd sums or differences

multiples of other integers

finite input constraints:

third digit is a 1
third-to-last digit is a 1
third digit == third-to-last digit
etc.

Uncomputable **with FSMs**

equal numbers of two values

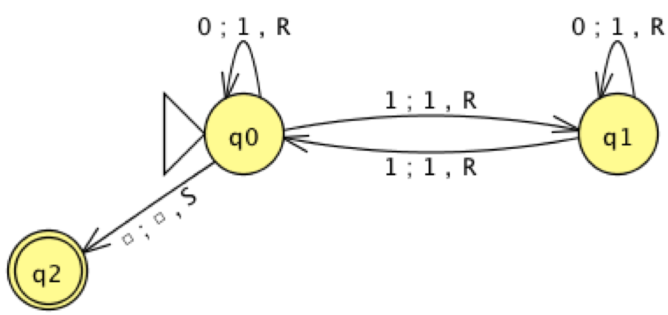
two more **1**s than **0**s or vice versa

infinite input constraints:

palindromes

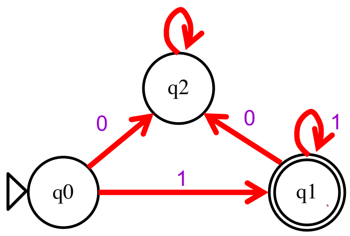
anything modeled by a potentially unbounded `while` *loop*

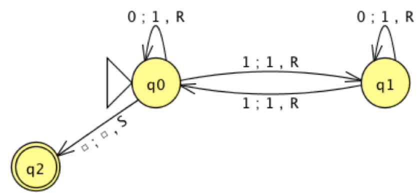
A Better Machine!



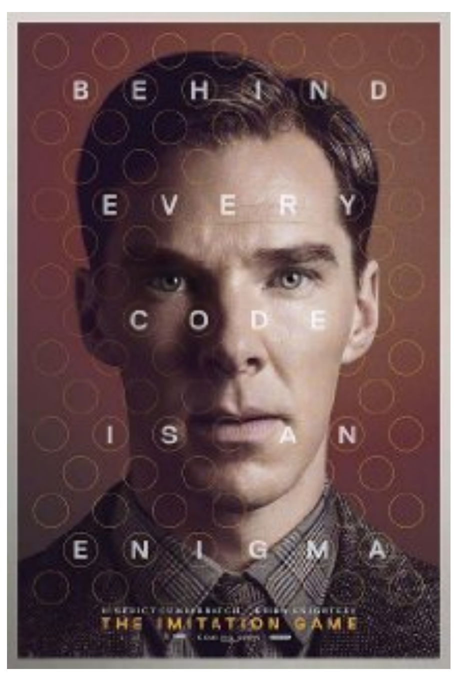
Turing Machine (TM)

FSM vs. TM

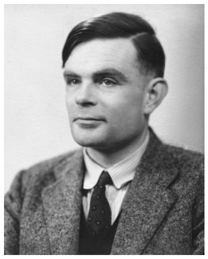




- A **finite state machine** is a restricted **Turing machine** where it can only perform "read" operations, and always moves from left to right.
- **Turing machines** have something more called **memory**



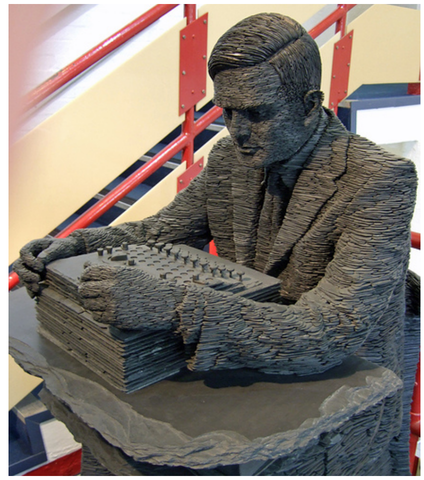
Alan Turing (1912-1954)



WWII

Enigma machine ~ The axis's encryption engine

1946

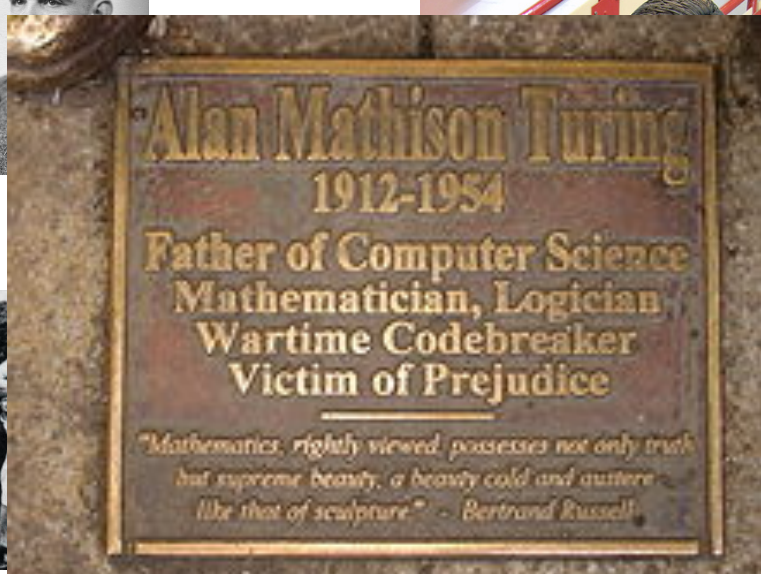


Bletchley Park

Turing Award

The ACM A.M. Turing Award is an annual prize given by the Association for Computing Machinery to "an individual selected for contributions of a technical nature made to the computing community". Wikipedia

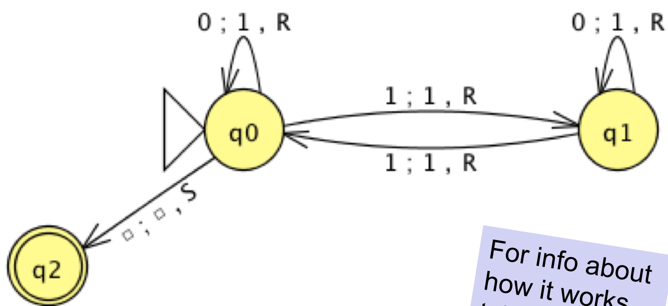
Alan Turing (1912-1954)



1946

computing community". Wikipedia

A Better Machine!



For info about how it works, take CS 332!

Turing Machine (TM)

So far, all known computational devices can compute <u>only</u> what Turing Machines can...

(but maybe faster...)

- Quantum computation
  http://www.cs.virginia.edu/~robins/The_Limits_of_Quantum_Computers.pdf
- Molecular computation
  http://www.arstechnica.com/reviews/2q00/dna/dna-1.html
- Parallel computers
- Integrated circuits

- Electromechanical computation
- Water-based computation
- Tinkertoy computation