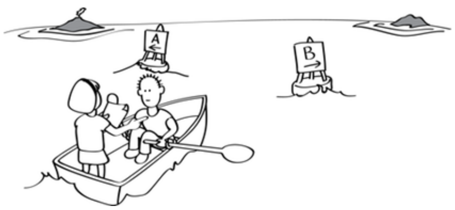
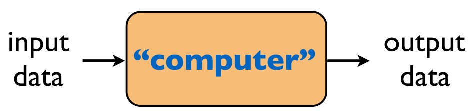
15-251
Great Ideas in
Theoretical Computer Science

Lecture 3:
Deterministic Finite Automaton (DFA), Part 1



September 5th, 2017

This Week and Next Week



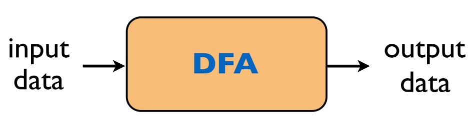
What is computation?

What is an algorithm?

How can we mathematically define them?

This Week

Introducing deterministic finite automata (DFA)



Let's assume two things about our world

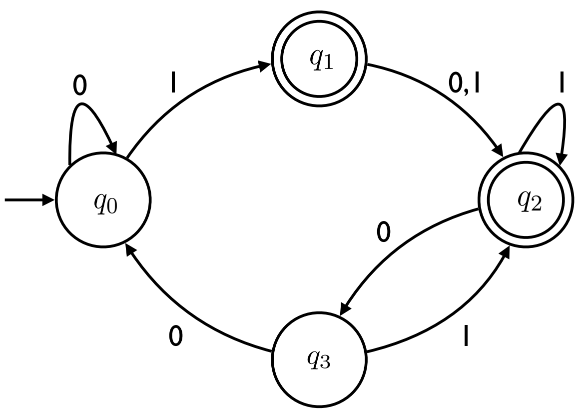
1. No universal machines exist.

+ | isPrime | Sorting

2. We only have machines to solve decision problems.

State diagram of a DFA

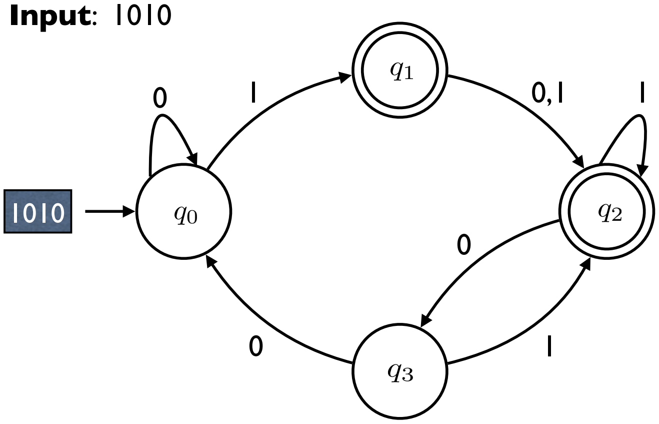
$\Sigma = \{0, 1\}$



Simulation of a DFA

$\Sigma = \{0, 1\}$

Input: 1010

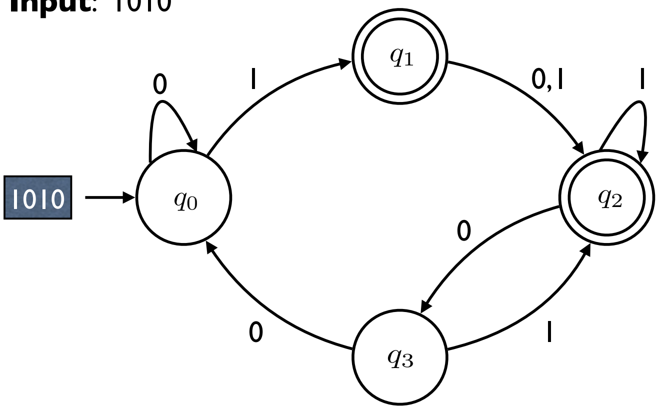


Simulation of a DFA

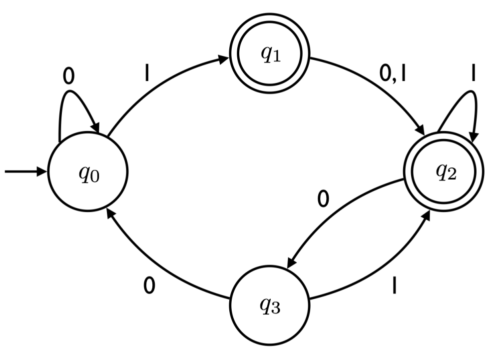
$\Sigma = \{0, 1\}$

Input: 1010

Decision: Reject



Anatomy of a DFA

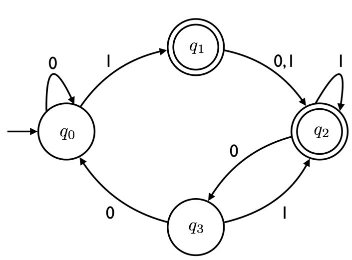


DFA as a programming language

```
def foo(input):
   i = 0;
   STATE 0:
      if (i == input.length): return False;
      letter = input[i];
      i++;
      switch(letter):
         case '0':  go to STATE 0;
         case '1':  go to STATE 1;

   STATE 1:
      if (i == input.length): return True;
      letter = input[i];
      i++;
      switch(letter):
         case '0':  go to STATE 2;
         case '1':  go to STATE 2;
   ...
```

input = | 0 | 1 | 1 | 1 | 1 |



Definition: Language decided by a DFA

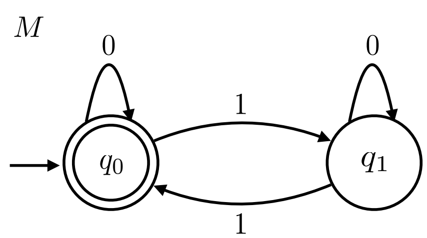
Let M be a DFA.

We let $L(M)$ denote the set of strings that M **accepts**.

So, $L(M) = \{x \in \Sigma^* : M(x) \text{ accepts.}\} \subseteq \Sigma^*$

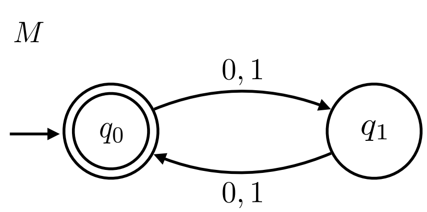
If $L = L(M)$, we say that M *recognizes* L.
accepts
decides
computes

DFA Examples



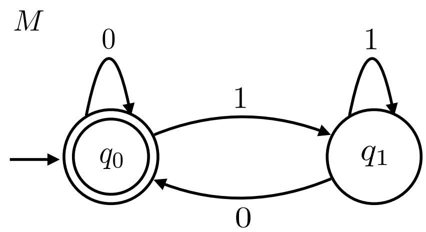
$L(M) =$

DFA Examples



$L(M) =$

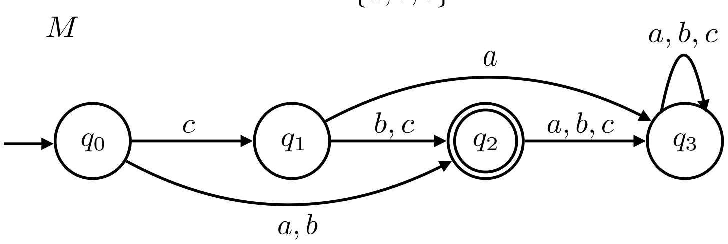
DFA Examples



$L(M) =$

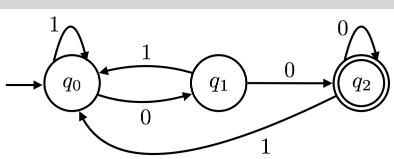
DFA Examples

$\Sigma = \{a, b, c\}$



$L(M) =$

Poll



The set of all words that contain at least three 0's
The set of all words that contain at least two 0's
The set of all words that contain 000 as a substring
The set of all words that contain 00 as a substring
The set of all words ending in 000
The set of all words ending in 00
The set of all words ending in 0
None of the above
Beats me

DFA construction practice

$L = \{110, 101\}$

$L = \{0,1\}^* \backslash \{110, 101\}$

$L = \{x \in \{0,1\}^* : x \text{ starts and ends with same bit.}\}$

$L = \{x \in \{0,1\}^* : |x| \text{ is divisible by 2 or 3.}\}$

$L = \{\epsilon, 110, 110110, 110110110, \ldots\}$

$L = \{x \in \{0,1\}^* : x \text{ contains the substring } 110.\}$

$L = \{x \in \{0,1\}^* : 10 \text{ and } 01 \text{ occur equally often in } x.\}$

Formal definition: DFA

A deterministic finite automaton (DFA) M is a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, F)$$

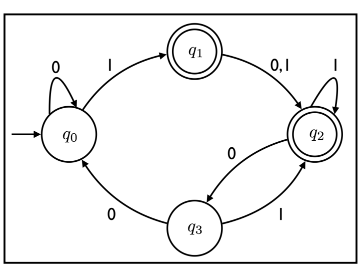
where

- Q
- Σ
- δ
- $q_0 \in Q$
- $F \subseteq Q$

Formal definition: DFA

A deterministic finite automaton (DFA) M is a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, F)$$



$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$\delta : Q \times \Sigma \to Q$

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_3	q_2
q_3	q_0	q_2

q_0 is the start state

$F = \{q_1, q_2\}$

Formal definition: DFA accepting a string

Let $w = w_1 w_2 \cdots w_n$ be a string over an alphabet Σ.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

We say that M ***accepts*** the string w
if there exists a sequence of states $r_0, r_1, \ldots, r_n \in Q$
such that

Otherwise we say M ***rejects*** the string w.

Formal definition: DFA accepting a string

Simplifying notation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

$\delta : Q \times \Sigma \rightarrow Q$ can be extended to $\delta^* : Q \times \Sigma^* \rightarrow Q$

as follows:

for $q \in Q, w \in \Sigma^*$,

$\delta^*(q, w) =$ state we end up in when we start at q and read w

In fact, even OK to drop $*$ from the notation.

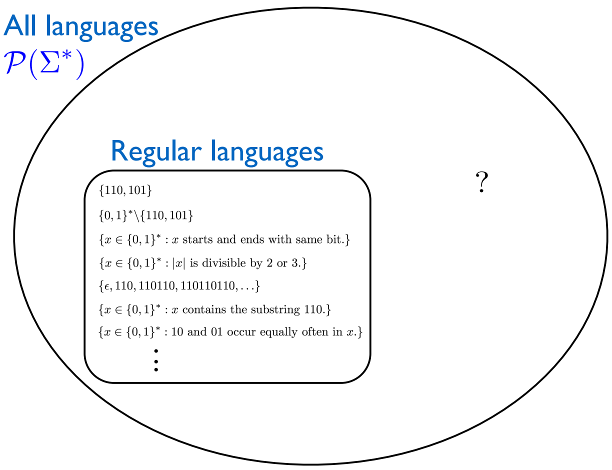
M ***accepts*** w if $\delta(q_0, w) \in F$.

Otherwise M ***rejects*** w.

Definition: Regular languages

Definition:

Regular languages



Regular languages

Questions:

1. Are all languages regular?
(Are all decision problems computable by a DFA?)

2. Are there other ways to tell if a language is regular?

A non-regular language

Theorem:
The language $L = \{0^n 1^n : n \in \mathbb{N}\}$ is **not** regular.

Note $L = \{\epsilon, 01, 0011, 000111, 00001111, \ldots\}$.

A non-regular language

Theorem:
The language $L = \{0^n 1^n : n \in \mathbb{N}\}$ is **not** regular.

Intuition:

A non-regular language

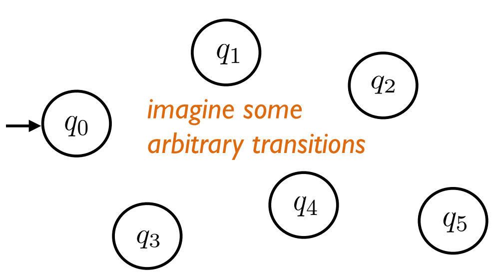
Theorem:
The language $L = \{0^n 1^n : n \in \mathbb{N}\}$ is **not** regular.

A key component of the proof:

A non-regular language

Warm-up:
Suppose a DFA with 6 states decides $L = \{0^n 1^n : n \in \mathbb{N}\}$.

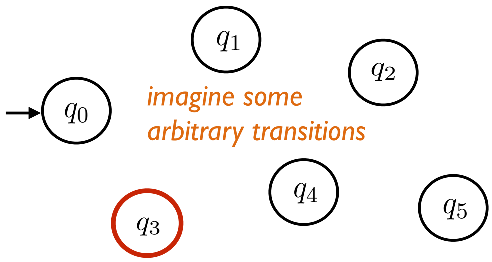
Input: 0000000011111111



A non-regular language

Warm-up:
Suppose a DFA with 6 states decides $L = \{0^n 1^n : n \in \mathbb{N}\}$.

Input: 0000000011111111



A non-regular language

Theorem:
The language $L = \{0^n 1^n : n \in \mathbb{N}\}$ is **not** regular.

Proof: Proof is by contradiction. So suppose L is regular.
This means there is a DFA M that decides L.
Let k denote the number of states of M.
Let r_n denote the state M is in after reading 0^n.
By PHP, there exists $i, j \in \{0, 1, \ldots, k\}$, $i \neq j$, such that $r_i = r_j$. So 0^i and 0^j end up in the same state.
For any string w, $0^i w$ and $0^j w$ end up in the same state.
But for $w = 1^i$, $0^i w$ should end up in an accepting state,
and $0^j w$ should end up in a rejecting state.
This is the desired contradiction. □

Proving a language is not regular

What makes the proof work:

Proving a language is not regular

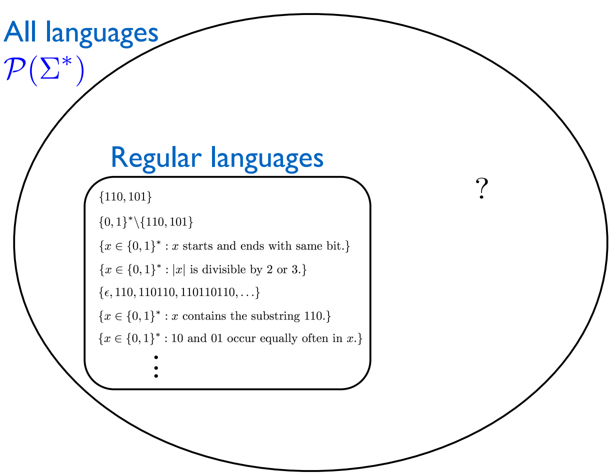
Exercise (test your understanding):

Show that the following language is not regular:

$$L = \{c^{251}a^n b^{2n} : n \in \mathbb{N}\}.$$

$(\Sigma = \{a, b, c\})$

Regular languages



Another non-regular language?

Question: Are all unary languages regular?

(a language L is unary if $L \subseteq \Sigma^*$, where $|\Sigma| = 1$.)

Theorem:
The language $\{a^{2^n} : n \in \mathbb{N}\}$ is **not** regular.

Regular languages

Questions:

1. Are all languages regular?
   (Are all decision problems computable by a DFA?)

2. Are there other ways to tell if a language is regular?

Next Time

Closure properties of regular languages