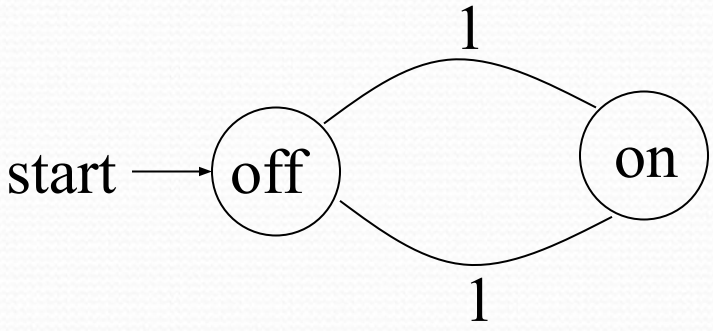
Topics

- Automata Theory
- Grammars and Languages
- Complexities

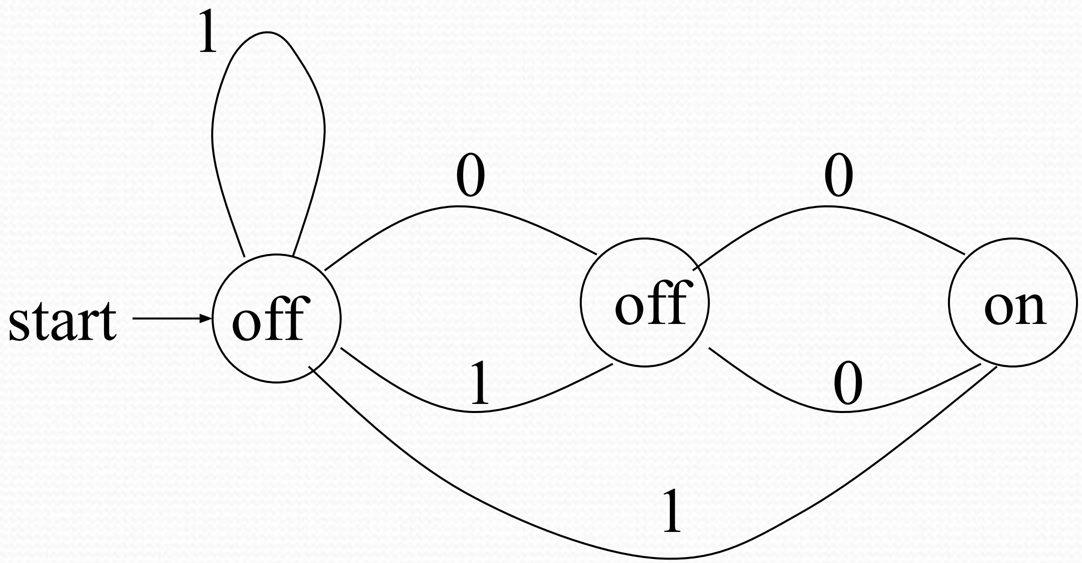
Why Automata Theory?

To study abstract computing devices which are closely related to today's computers. A simple example of *finite state machine*:



There are many different kinds of machines.

Another Example



When will this be *on*?
Try 100, 1001, 1000, 111, 00, …

Grammar and Languages

Grammars and languages are closely related to automata theory and are the basis of many important software components like:

- Compilers and interpreters
- Text editors and processors
- Search engines
- System verification components

Complexities

Study the limits of computations. What kinds of problems can be solved with a computer? What kinds of problems can be solved *efficiently*?

Preliminaries

- Alphabets
- Strings
- Languages
- Problems

Alphabets

- An alphabet is a finite set of symbols.
- Usually, use Σ to represent an alphabet.
- Examples:
  - $\Sigma = \{0,1\}$, the set of binary digits.
  - $\Sigma = \{a, b, \ldots, z\}$, the set of all lower-case letters.
  - $\Sigma = \{(,)\}$, the set of open and close parentheses.

Strings

- A string is a finite sequence of symbols from an alphabet.
- Examples:
  - 0011 and 11 are strings from $\Sigma = \{0,1\}$
  - abc and bbb are strings from $\Sigma = \{a, b, \ldots, z\}$
  - (()(())) and)(() are strings from $\Sigma = \{(,)\}$

Strings

- **Empty string**: ε
- **Length** of string: |0010| = 4, |aa| = 2, |ε|=0
- **Prefix** of string: <u>aa</u>abc, <u>aaab</u>c, <u>aaabc</u>
- **Proper prefix** of string: <u>aa</u>abc, <u>aaab</u>c
- **Suffix** of string: aaa<u>bc</u>, aa<u>abc</u>, <u>aaabc</u>
- **Proper suffix** of string: aaa<u>bc</u>, aa<u>abc</u>
- **Substring** of string: aa<u>ab</u>c, <u>aaa</u>bc, aaab<u>c</u>

Strings

- **Concatenation**: ω=abd, α=ce, $\omega\alpha$=abdce
- **Exponentiation**: ω=abd, ω^3=abdabdabd, $\omega^0=\varepsilon$
- **Reversal**: ω=abd, ω^R = dba
- Σ^k = set of all k-length strings formed by symbols in Σ
  - e.g., $\Sigma=\{a,b\}$, $\Sigma^2=\{ab, ba, aa, bb\}$, $\Sigma^0=\{\varepsilon\}$

What is Σ^1? Is Σ^1 different from Σ? How?

Strings

- **Kleene Closure** $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots = \bigcup_{k\geq 0} \Sigma^k$

  e.g., $\Sigma=\{a, b\}$, $\Sigma^* = \{\varepsilon, a, b, ab, aa, ba, bb, aaa, aab, abb, \ldots\}$ is the set of all strings formed by a's and b's.

- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots = \bigcup_{k>0} \Sigma^k$
  - i.e., Σ^* without the empty string.

Languages

- A language is a set of strings over an alphabet.
- Examples:
  - $\Sigma=\{(,)\}$, $L_1=\{(),)(, (())\}$ is a language over Σ.
  - $\Sigma=\{a, b, c, \dots, z\}$, the set L of all legal English words is a language over Σ.
  - The set $\{\varepsilon\}$ is a language over any alphabet.

What is the difference between ϕ and $\{\varepsilon\}$?

Languages

- Other Examples:
  - $\Sigma=\{0, 1\}$, $L=\{0^n1^n \mid n\geq 1\}$ is a language over Σ consisting of the strings {01, 0011, 000111, … }
  - $\Sigma=\{0, 1\}$, $L = \{0^i1^j \mid j\geq i\geq 0\}$ is a language over Σ consisting of the strings with some 0's (possibly none) followed by at least as many 1's.

Problems

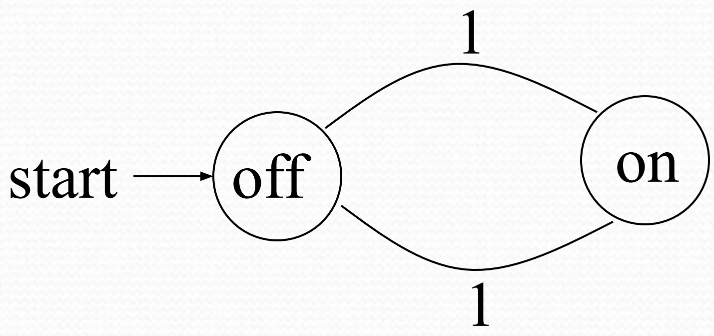
- In automata theory, a problem is to decide whether a given string is a member of some particular language.
- This formulation is general enough to capture the difficulty levels of all problems.

Finite Automata (or Finite State Machines)

- This is the simplest kind of machine.
- We will study 3 types of Finite Automata:
  - Deterministic Finite Automata (DFA)
  - Non-deterministic Finite Automata (NFA)
  - Finite Automata with ε-transitions (ε-NFA)

Deterministic Finite Automata (DFA)

- We have seen a simple example before:



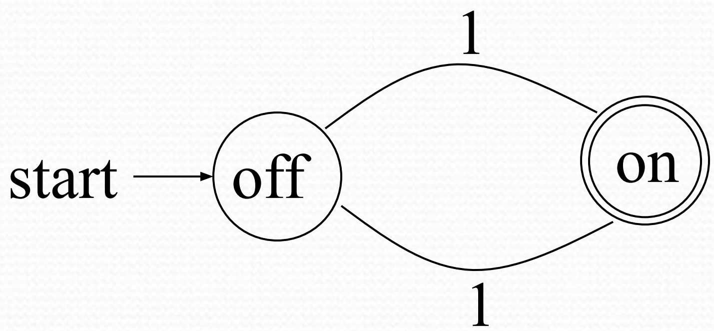
There are some <u>states</u> and <u>transitions</u> (edges) between the states. The edge labels tell when we can move from one state to another.

Definition of DFA

- A DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
  - Q is a finite set of <u>states</u>
  - Σ is a finite input <u>alphabet</u>
  - δ is the <u>transition function</u> mapping $Q \times \Sigma$ to Q
  - q_0 in Q is the <u>initial state</u> (only one)
  - $F \subseteq Q$ is a set of <u>final states</u> (zero or more)

Definition of DFA

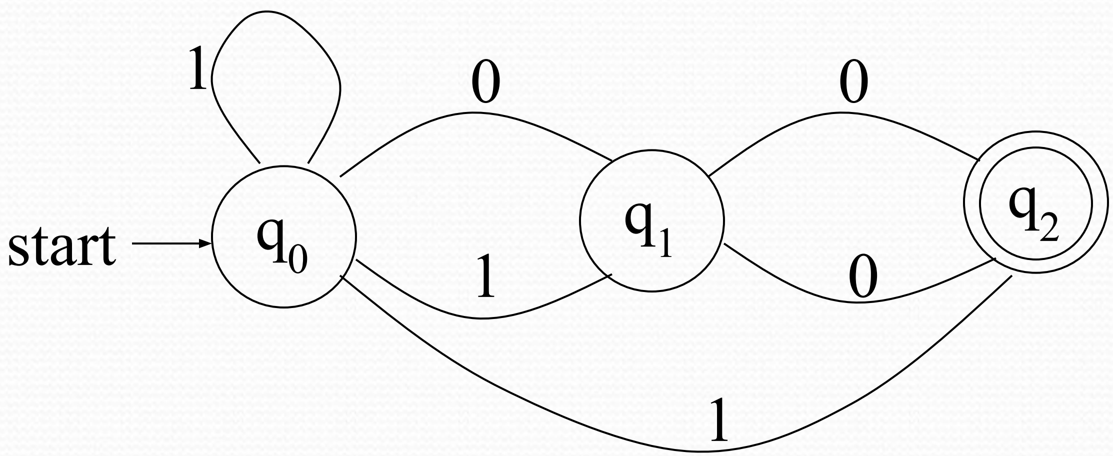
- For example:



Q is the set of states: {on, off}
Σ is the set of input symbols: {1}
δ is the transitions: off $\times$ 1 $\rightarrow$ on; on $\times$ 1 $\rightarrow$ off
q_0 is the initial state: off
F is the set of final states (double circle): {on}

Definition of DFA

- Another Example:



What are Q, Σ, δ, q_0 and F in this DFA?

Transition Table

- For the previous example, the DFA is (Q,Σ,δ,q_0,F) where $Q = \{q_0,q_1,q_2\}$, $\Sigma = \{0,1\}$, $F = \{q_2\}$ and δ is such that

States	Inputs	
	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
$*q_2$	q_1	q_0

Note that there is <u>one transition only</u> for each input symbol from each state.

Language of a DFA

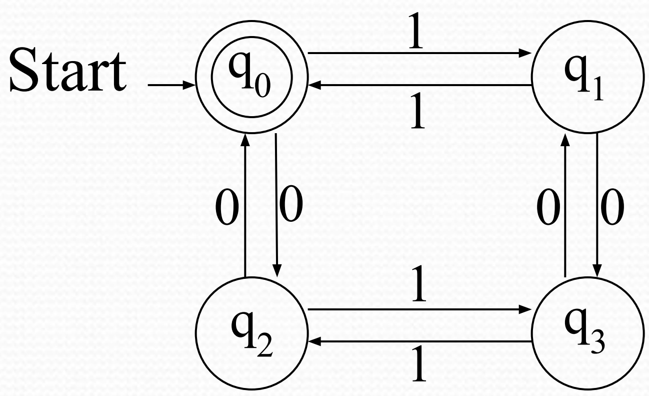
- Given a DFA M, the language accepted (or recognized) by M is the set of all strings that, starting from the initial state, will reach one of the final states after the whole string is read.
- For example, the language accepted by the previous example is the string that ends with oo

DFA Example

- Consider the DFA $M=(Q,\Sigma,\delta,q_o,F)$ where $Q = \{q_o,q_1,q_2,q_3\}$, $\Sigma = \{0,1\}$, $F = \{q_o\}$ and δ is:

States	Inputs 0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

OR



We can use a <u>transition table</u> or a <u>transition diagram</u> to specify the transitions. What input can take you to the final state in M?