16.070

Introduction to Computers & Programming

Theory of computation: What is a computer? FSM, Automata

Prof. Kristina Lundqvist
Dept. of Aero/Astro, MIT

Models of Computation

What is a computer?

- If you can’t measure it it has no value…
  - Quantitative (numerical)
  - Qualitative
- Can we model a computer as we know it today?

Models of Computation

Uncomputable	
Turing Machines	Phrase Structure
Linear bounded automata	Context-sensitive
Pushdown automata	Context-free
Finite state automata	Regular
Machines	Grammars/Languages

Complex

Crude

Finite Machines

- Think of a ***black box*** which takes inputs from the environment and produces some kind of observable response. Examples are e.g., vending machine, dish washer, automatic door opener
- A finite machine has a finite memory. It can only distinguish between a finite number of input ***histories***.
- Each class of equivalent histories corresponds to a ***state*** of the machine.

A Finite State Automata (FSA) is an abstract finite machine

Finite Automata

(Merriam-Webster)

One entry found for **automaton.**

Main Entry: **au·tom·a·ton**
Pronunciation: o-'tä-m&-t&n, -m&-"tän
Function: *noun*
Inflected Form(s): *plural* **-atons** *or* **au·tom·a·ta** /-m&-t&, -m&-"tä/
Etymology: Latin, from Greek, neuter of *automatos*
Date: 1645
1 : a mechanism that is relatively self-operating; *especially* **: ROBOT**
2 : a machine or control mechanism designed to follow automatically a predetermined sequence of operations or respond to encoded instructions
3 : an individual who acts in a mechanical fashion

Theory of Computation

1. **Finite state automata**: deterministic and non-deterministic state machines, regular expressions and languages. Techniques for identifying and describing regular languages; techniques for showing that a language is not regular. Properties of such languages.

2. **Context-free languages**: Context-free grammars, parse trees, derivations and ambiguity. Relation to pushdown automata. Properties of such languages and techniques for showing that a language is not context-free.

Theory of Computation

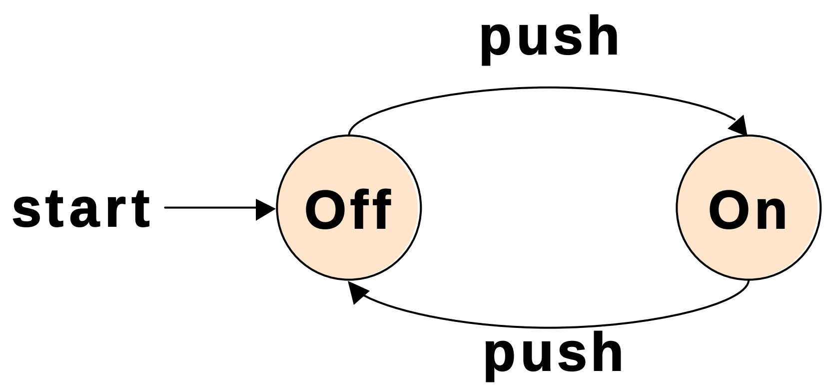
3. **Turing Machines**: Basic definitions and relation to the notion of an algorithm or program. Power of Turing Machines.

4. **Undecidability**: Recursive and recursively enumerable languages. Universal Turing Machines. Power of Turing Machines.

5. **Computational Complexity**: Decidable problems for which no sufficient algorithms are known. Polynomial time computability. The notion of NP-completeness and problem reductions. Example of hard problems.

Finite Automata

- A simple finite automaton; an on/off-switch



Initially the switch is Off. When the switch encounters a Push it changes state into the On state. When another Push is encountered we switch into the Off state and so on.

  - Circles represent **states**. In this case named *On* and *Off*.
  - Edges (arcs) represent **transitions** or **input** to the system.
  - Start arrow indicates which state we start in

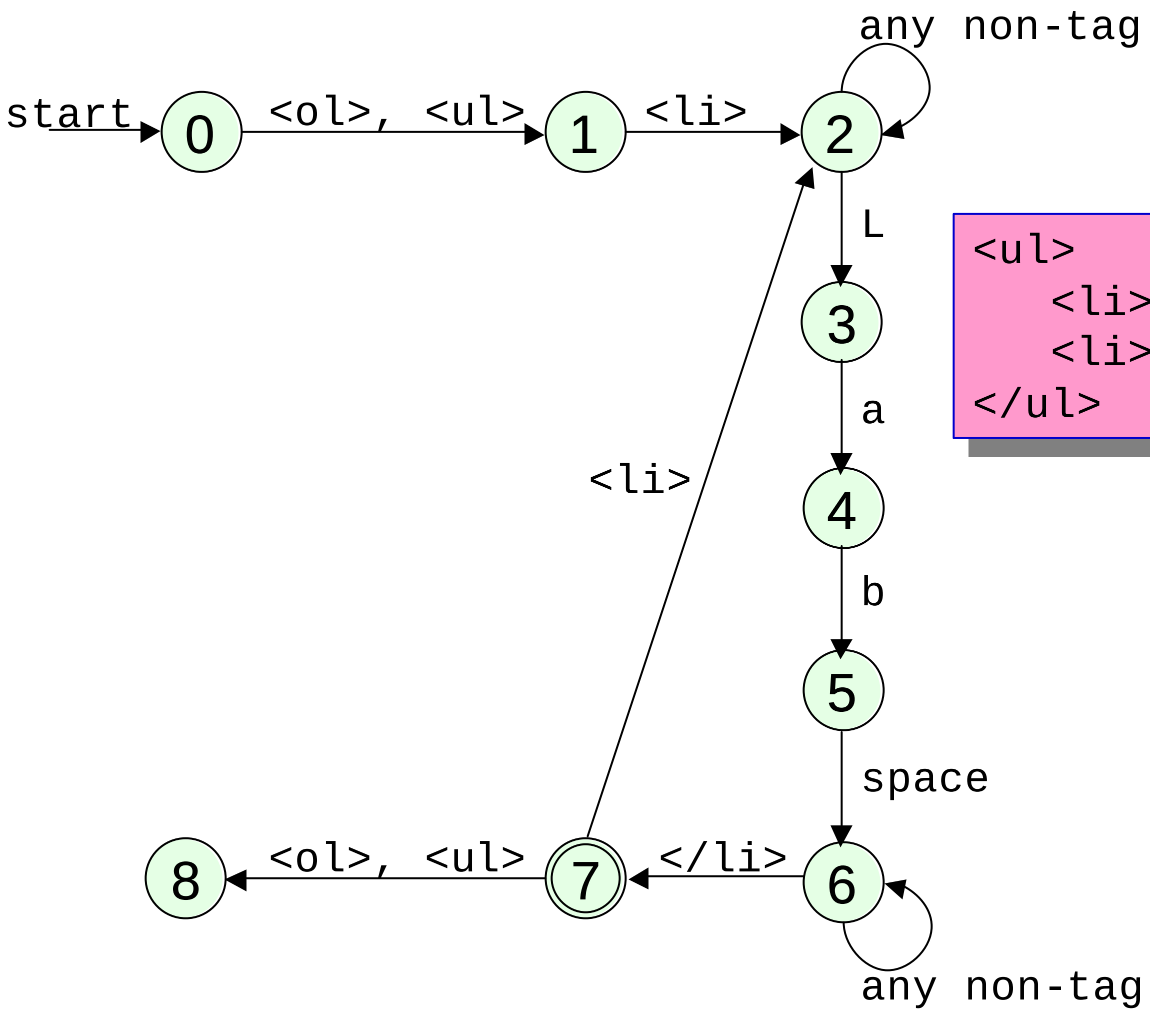
Finite Automata

- Software to design and verify circuit behavior
- Lexical analyzer of a typical compiler
- Parser for natural language processing
- An efficient scanner for patterns in large bodies of text (e.g. text search on the web)
- Verification of protocols (e.g. communications, security).
- …

Moore and Mealy Machines

- Two types of machines: **Moore** and **Mealy**. The difference lies in the outputs.
- Mealy Machines
  - The output is a function of the present state **and** all the inputs
  - Input change causes an immediate output change
- Moore Machines
  - The output is a function of the present state only
  - Outputs change synchronously with state changes

Finite Automata



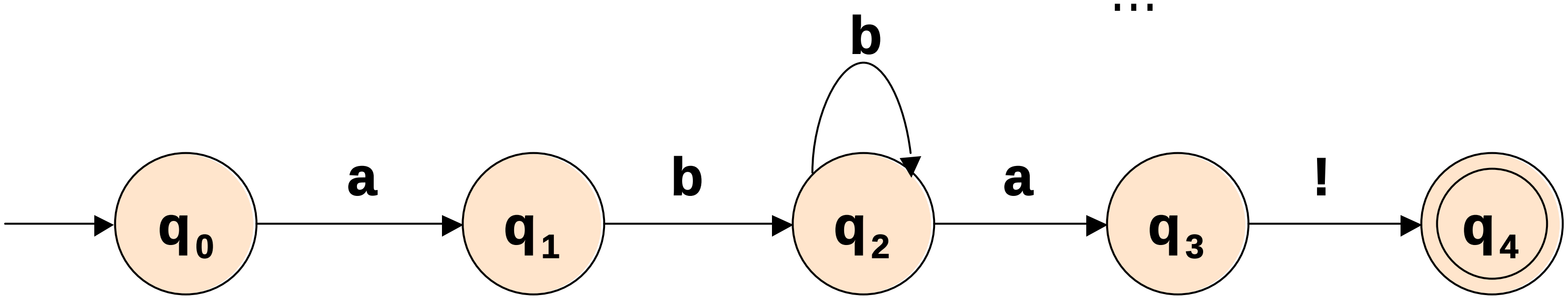

```
<ul>
   <li>Lab 1 </li>
   <li>Lab 2 </li>
</ul>
```

- Lab 1
- Lab 2

Finite State Automata

abba! → **Accept** / **Reject**

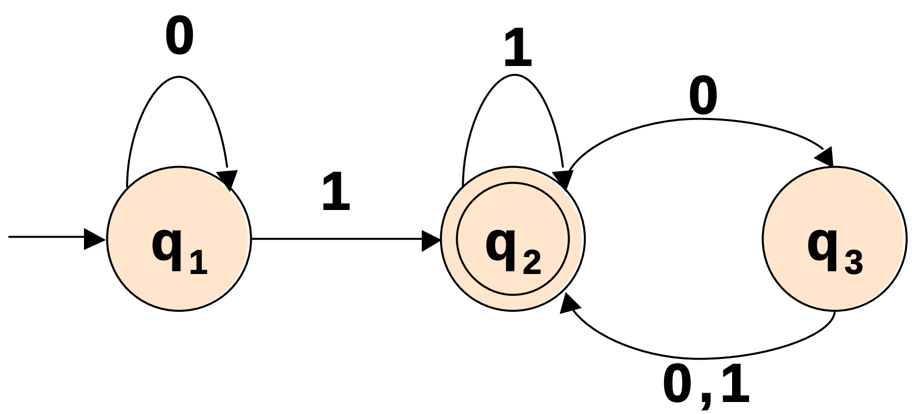
aba!
abba!
abbba!
...
⇒ ab+a!



A finite automaton called M_1
The figure is called the ***state diagram*** of M_1
It has 5 ***states*** labeled q_0, q_1, q_2, q_3, q_4,
The ***start state*** is labeled q_0
The ***accept state*** q_2, is the one with double circles
The arrows going from one state to another are called ***transitions***

Formal Definition of a Finite Automaton

- An FSA 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**
  5. $F \subseteq Q$ is the **set of accept states** (final states)



$Q = \{q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

q1 is the start state
$F = \{q_2\}$

Formal Definition of a Finite Automaton

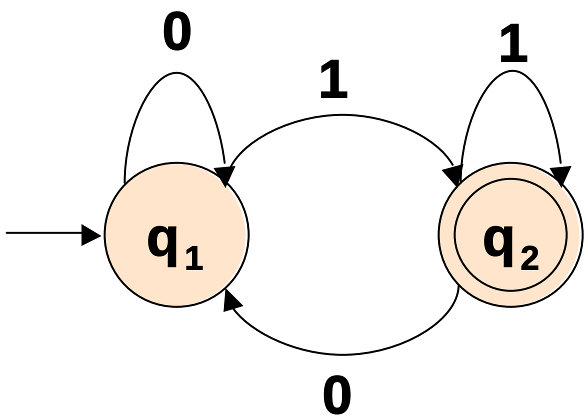
- If A is the set of all strings that machine M accepts, we say that A is the **language of machine M** and write L(M)=A

- We say that **M recognizes A** (or that **M accepts A**.)

- A machine may accept several strings, but it only recognizes **one** language.

Finite Automaton M_2

- State diagram of finite automaton M_2



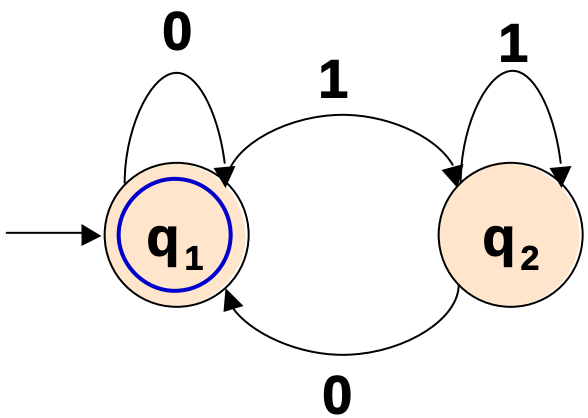
$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

What strings does M_2 accept?

Finite Automaton M_3

- State diagram of finite automaton M_3



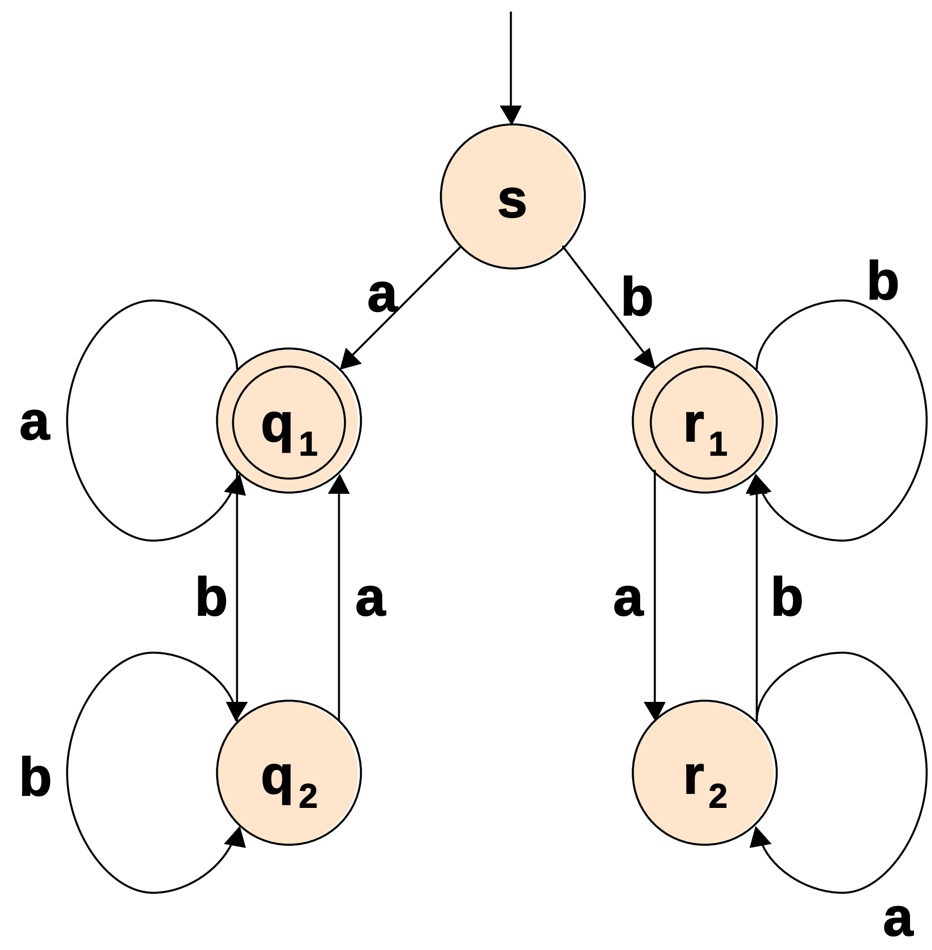
$L(M_3) = \{\omega \mid \omega$ is the empty string ϵ or ends in a $0\}$.

Finite Automaton M_4

Alphabet Σ = {a, b}

What does M_4 accept?

All strings that start and end with a, or that start and end with b. In other words, M4 accepts strings that start and end with the same symbol.

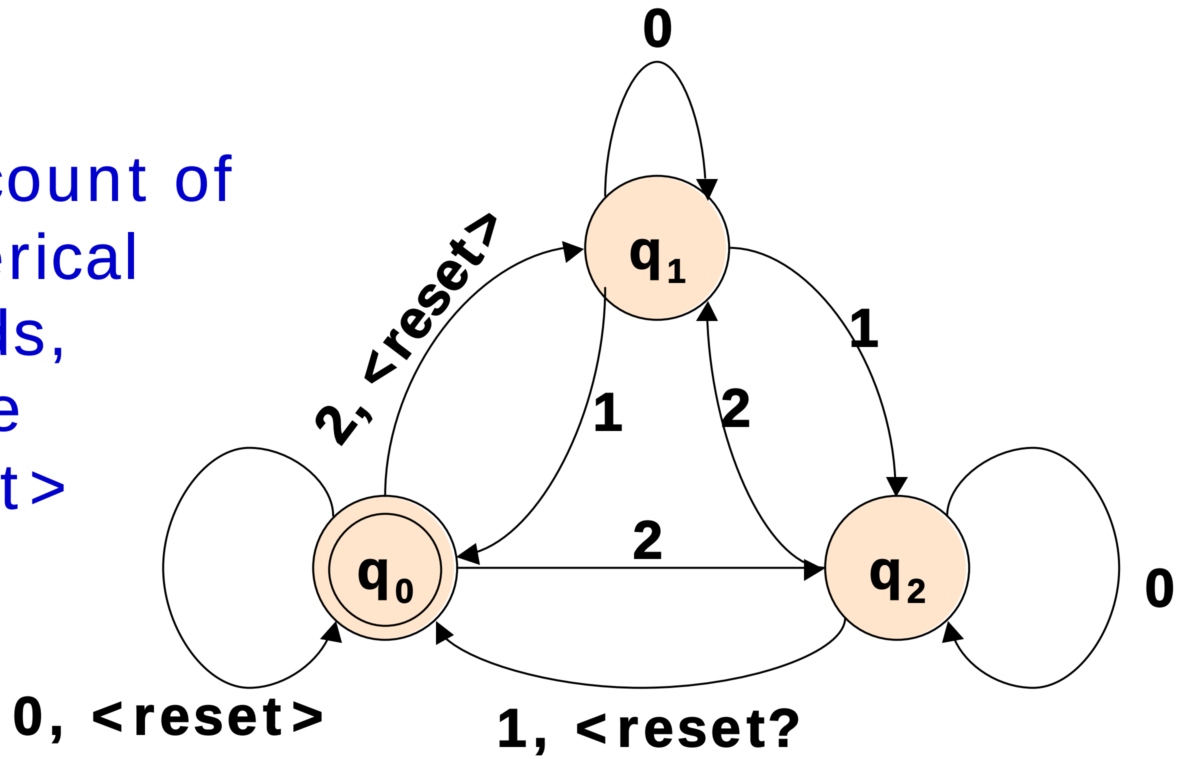


Finite Automaton M_5

Alphabet Σ = {<reset>, 0, 1, 2}

What does M_5 accept?

M_5 keeps a running count of the sum of the numerical input symbols it reads, modulo 3. Every time it receives the <reset> symbol it resets the count to 0.



Finite Automaton M_6

- Is it possible to describe all finite automata by a state diagram?
- No: if diagram is to large to draw
- No: if description depends on some unspecified parameter

$$B_i = (Q_i, \Sigma, \delta_i, q_0, \{q_0\})$$

Formal Definition of Computation

- Let $M = (Q, \Sigma, \delta_i, q_0, F)$ be a finite automaton
- Let $w = w_1w_2\ldots w_n$ be a string over the alphabet Σ

- Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in Q with the following three conditions:
  1. $r_0 = q_0$
  2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$
  3. $r_n \in F$

- M recognizes language A if $A = \{w \mid M \text{ accepts } w\}$

Regular Language

- A language is called a **regular language** if some finite automaton recognizes it.