Formal Language and Automata Theory

Lecture 1: Introduction

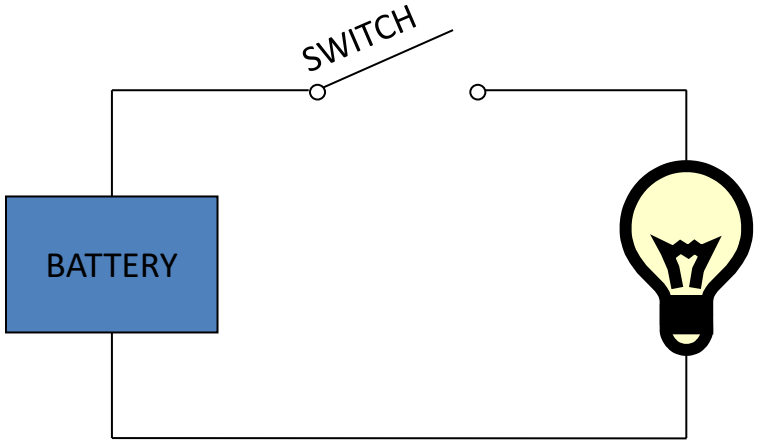
Why Study Automata?

- A survey of Stanford grads 5 years out asked which of their courses did they use in their job.
- Basics like CS106 (Programming courses) took the top spots, of course.
- But among optional courses, CS154 (Introduction to Automata and Complexity Theory) stood remarkably high.
- One of the most fundamental courses of Computer Science.
- It is mainly about what kind of things can really compute mechanically.

What is automata theory

- Automata theory is the study of abstract computational devices
- Abstract devices are (simplified) models of real computations
- Computations happen everywhere: On your laptop, on your cell phone, in nature, …
- Why do we need abstract models?

Finite automata model protocols, electronic circuits, …



Theory is used in *model-checking*

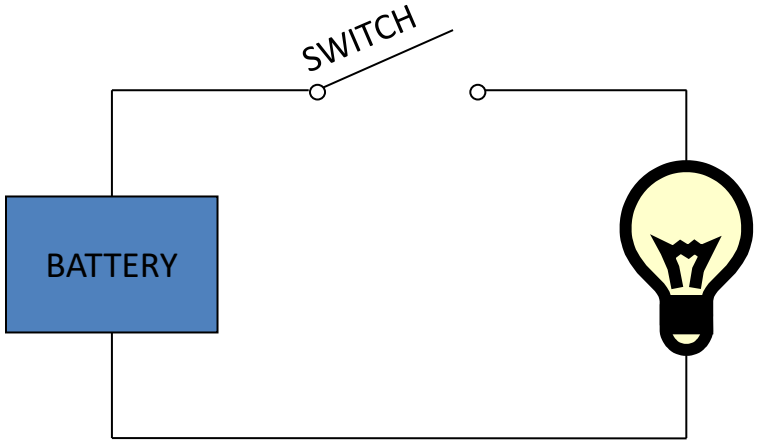
input: switch

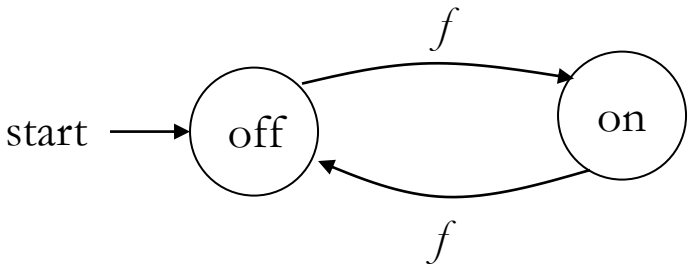
output: light bulb

actions: flip switch

states: on, off

A simple "computer"





input: switch

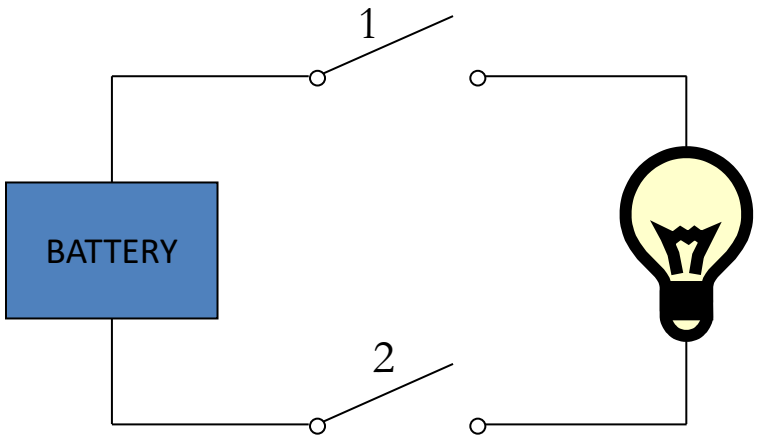
output: light bulb

actions: *f* for "flip switch"

states: on, off

bulb is on if and only if there was an odd number of flips

Another "computer"

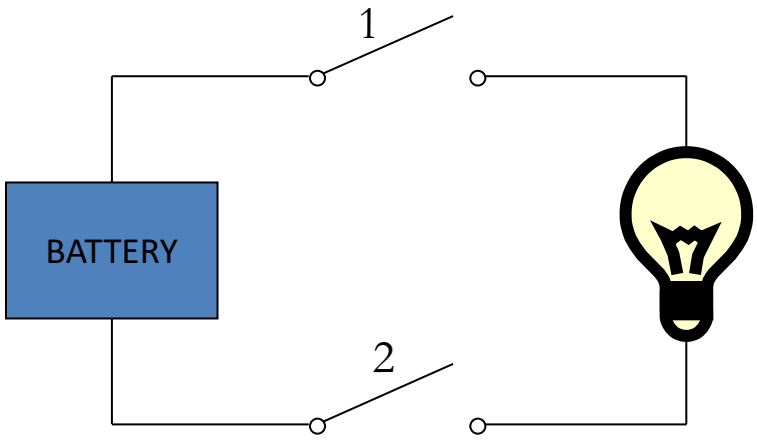


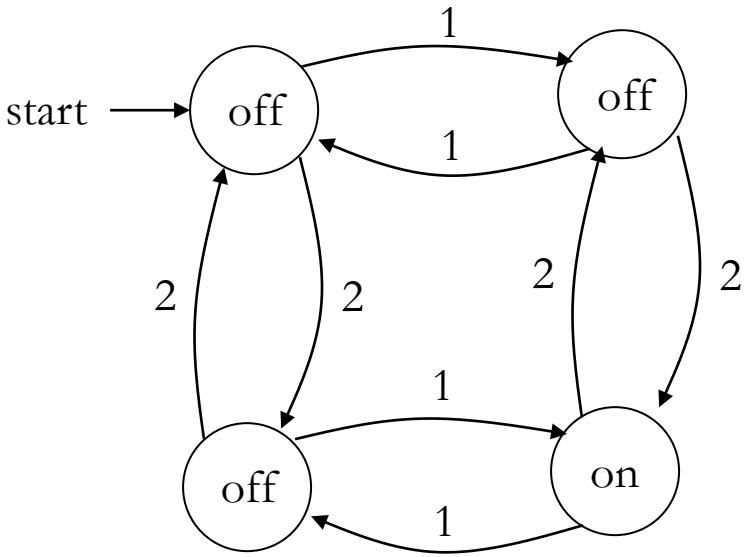
inputs: switches 1 and 2

actions: 1 for "flip switch 1"
2 for "flip switch 2"

states: on, off

Another "computer"





inputs: switches 1 and 2

actions: 1 for "flip switch 1"
2 for "flip switch 2"

states: on, off

bulb is on if and only if
both switches were flipped
an odd number of times

What kind of things can really compute mechanically?

Question:

Do you know how a vending machine works? Can you design one?

Vending machine room seen in Hokkaido, Japan 2004

- How to design a vending machine?

  $\rightarrow$ *Use a finite automaton!*

  - Assumptions (for simplicity):

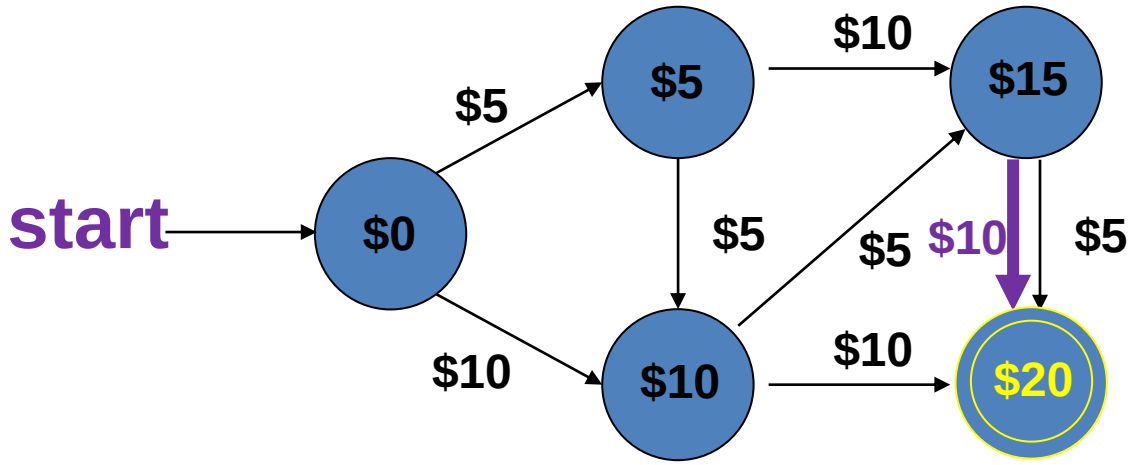
    – Only 5-dollar and 10-dollar coins are used.

    – Only drinks all of 20 dollars are sold.

Only 5-dollar and 10-dollar coins are used.
Only drinks all of 20 dollars are sold.

Solution

Requiring “memory” called “states” for the design.



transition diagram

A gumball machine

machine takes $5 and $10 coins

a gumball costs $15

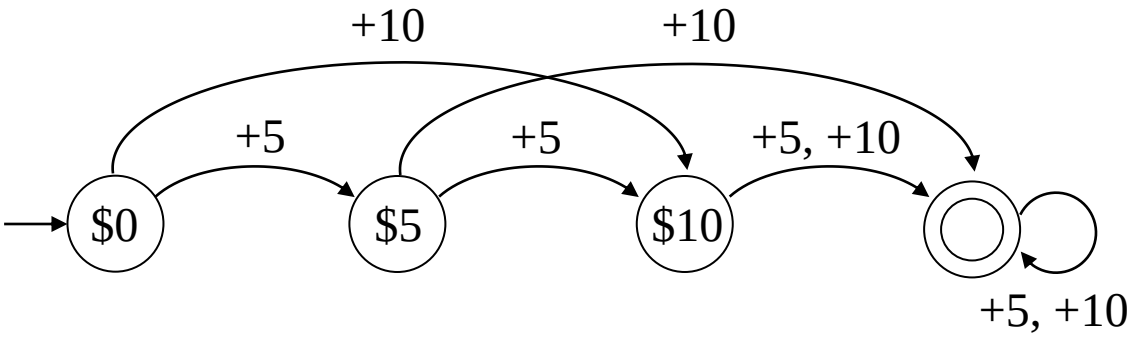
actions: +5, +10

A gumball machine

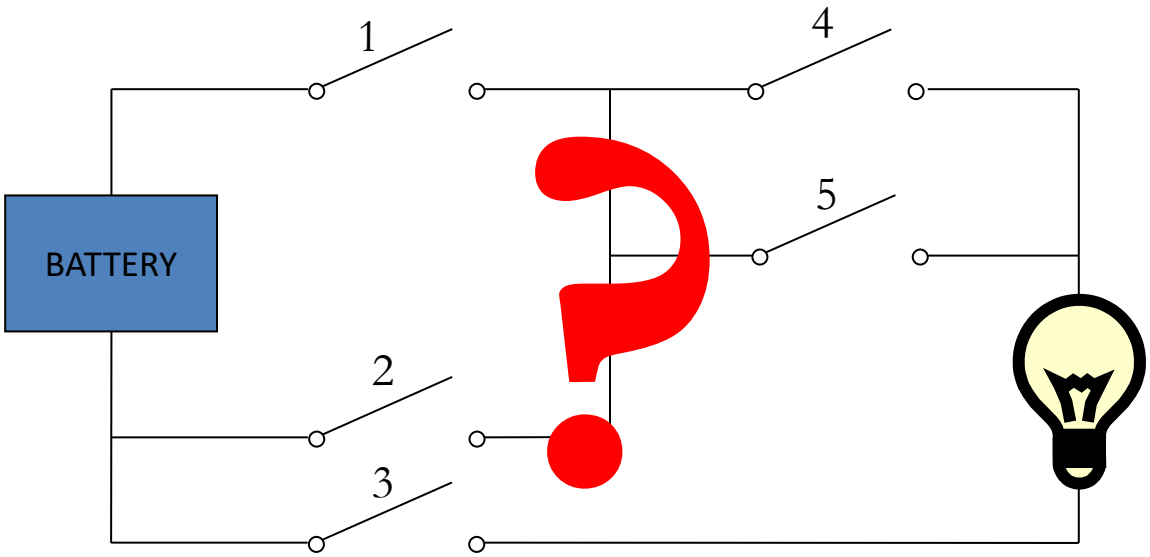
machine takes $5 and $10 coins

a gumball costs $15

actions: +5, +10



A design problem



Can you design a circuit where the light is on if and only if all the switches were flipped exactly the same number of times?

A design problem

- Such devices are difficult to reason about, because they can be designed in an infinite number of ways
- By representing them as abstract computational devices, or automata, we will learn how to answer such questions

These devices can model many things

- They can describe the operation of any “small computer”, like the control component of an alarm clock or a microwave
- They are also used in lexical analyzers to recognize well formed expressions in programming languages:

  `ab1` is a legal name of a variable in C

  `5u=` is not

Languages & Grammars

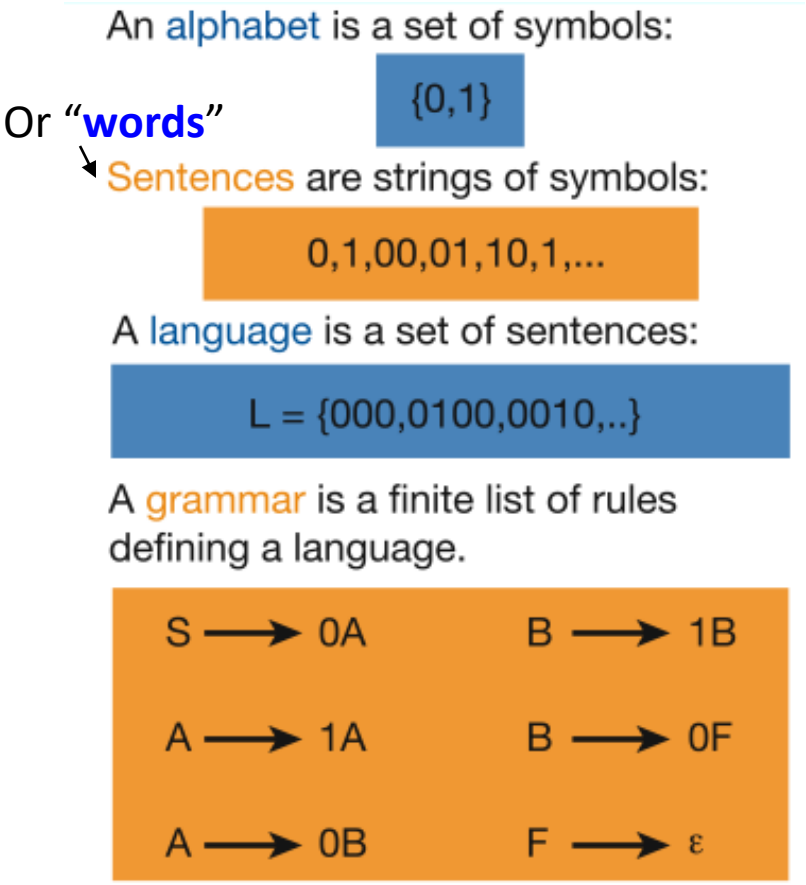


Image source: Nowak et al. Nature, vol 417, 2002

- Languages: "*A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols*"
- Grammars: "*A grammar can be regarded as a device that enumerates the sentences of a language*" - nothing more, nothing less

- *N. Chomsky, Information and Control, Vol 2, 1959*

Context-free grammars

- They are used to describe the syntax of essentially every programming language.
  - Not to forget their important role in describing natural languages.

Some devices we will see

finite automata	Devices with a finite amount of memory. Used to model "small" computers.
push-down automata	Devices with infinite memory that can be accessed in a restricted way. Used to model parsers, etc.
Turing Machines	Devices with infinite memory. Used to model any computer.

Turing Machines

- This is a general model of a computer, capturing anything we could ever hope to compute
- Surprisingly, there are many things that we cannot compute, for example:

> Write a program that, given the code of another program in C, tells if this program ever outputs the word "`hello`"

- It seems that you should be able to tell just by looking at the program, but it is impossible to do!

Problems

- Examples of problems we will consider
  - Given a word s, does it contain the subword "fool"?
  - Given a number n, is it divisible by 7?
  - Given a pair of words s and t, are they the same?
  - Given an expression with brackets, e.g. (( ) ( )), does every left bracket match with a subsequent right bracket?
- All of these have "yes/no" answers.

Some highlights of the course

- Finite automata
  - Automata are closely related to the task of searching for patterns in text

```
find (ab)*(ab) in abracadabra
```

- Grammars
  - Grammars describe the meaning of sentences in English, and the meaning of programs in Java
  - We will see how to extract the meaning out of a program

Theory of computations layers