Lesson 1: Formal Languages and Automata

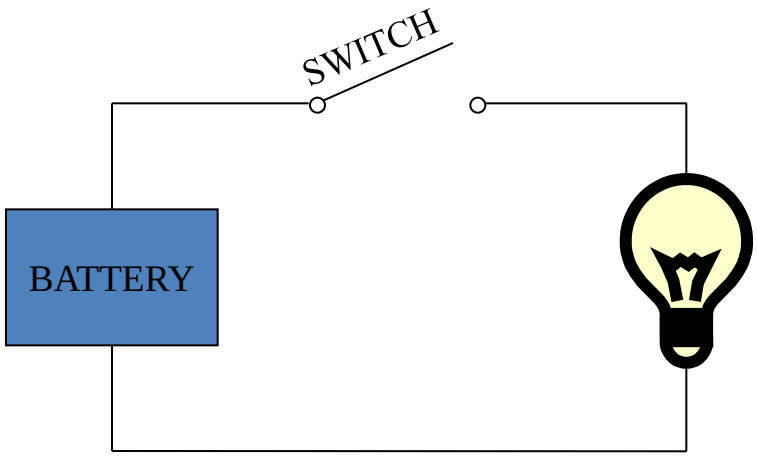
Marc Gaetano

Edition 2018

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?

A simple computer



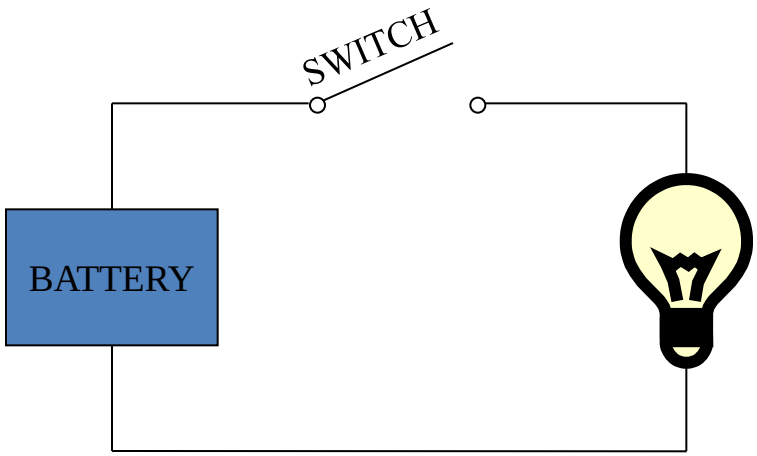
input: switch

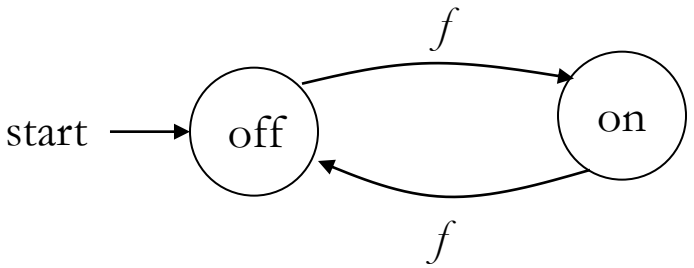
output: light bulb

actions: flip switch

states: on, off

A simple "computer"





input: switch

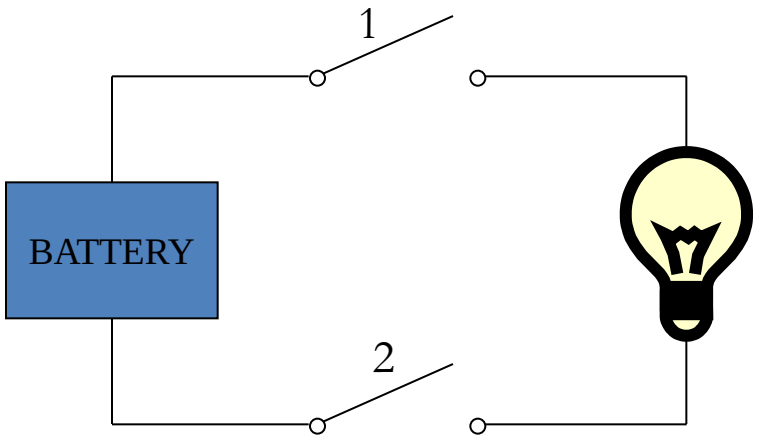
output: light bulb

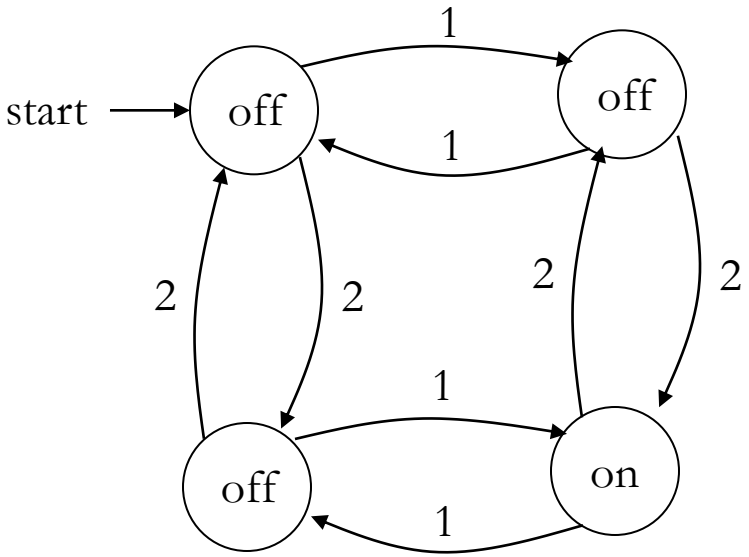
actions: f for "flip switch"

states: on, off

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

Another "computer"





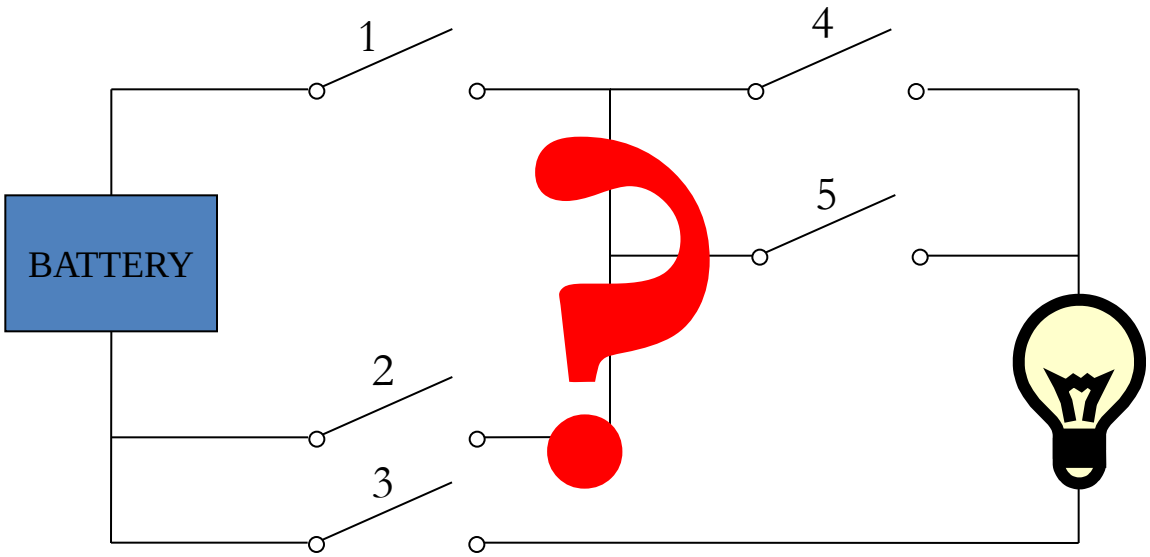
inputs: switches 1 and 2

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

states: on, off

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

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

Different kinds of automata

- This was only one example of a computational device, and there are others
- We will look at different devices, and look at the following questions:
  - What can a given type of device compute, and what are its limitations?
  - Is one type of device more powerful than another?

Some devices

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.
time-bounded Turing Machines	Infinite memory, but bounded running time. Used to model any computer program that runs in a "reasonable" amount of time.

Some highlights of the course

- Finite automata
  - We will understand what kinds of things a device with finite memory can do, and what it cannot do
  - Introduce simulation: the ability of one device to "imitate" another device
  - Introduce nondeterminism: the ability of a device to make arbitrary choices
- Push-down automata
  - These devices are related to grammars, which describe the structure of programming (and natural) languages

Some highlights of the course

- 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!

Preliminaries of automata theory

- How do we formalize the question

Can device A solve problem B?

- First, we need a formal way of describing the problems that we are interested in solving

Problems

- Examples of problems we will consider
  - Given a word, does it contain a given subword?
  - Given a number n, is it divisible by 7?
  - Given an expression with brackets, e.g. (( ) ( )), does every left bracket match with a subsequent right bracket?
- All of these have "yes/no" answers.
- There are other types of problems, that ask "Find this" or "How many of that" but we won't look at those.