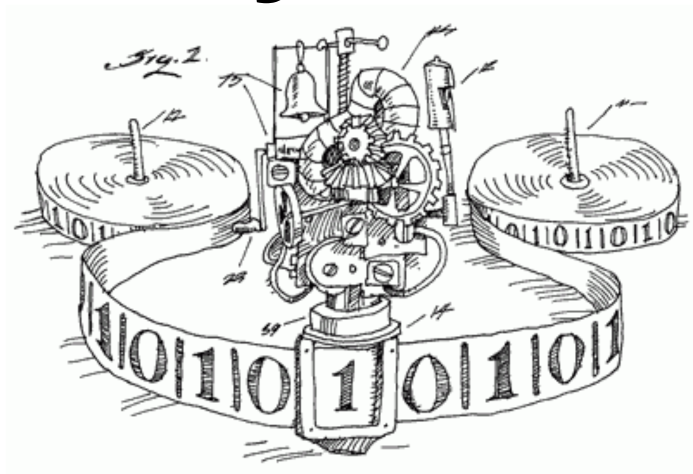
INFO 101 – Introduction to Computing and Security



[2020 - Week 9 / 1]

Prof. Dr. Rui Abreu

University of Porto, Portugal

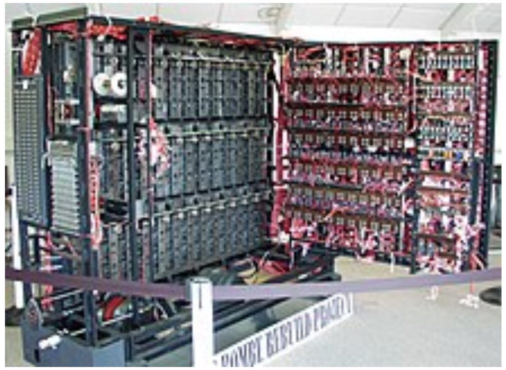
rui@computer.org

@rmaranhao

What we are going to discuss…

- Introducing computational principles: Turing machines

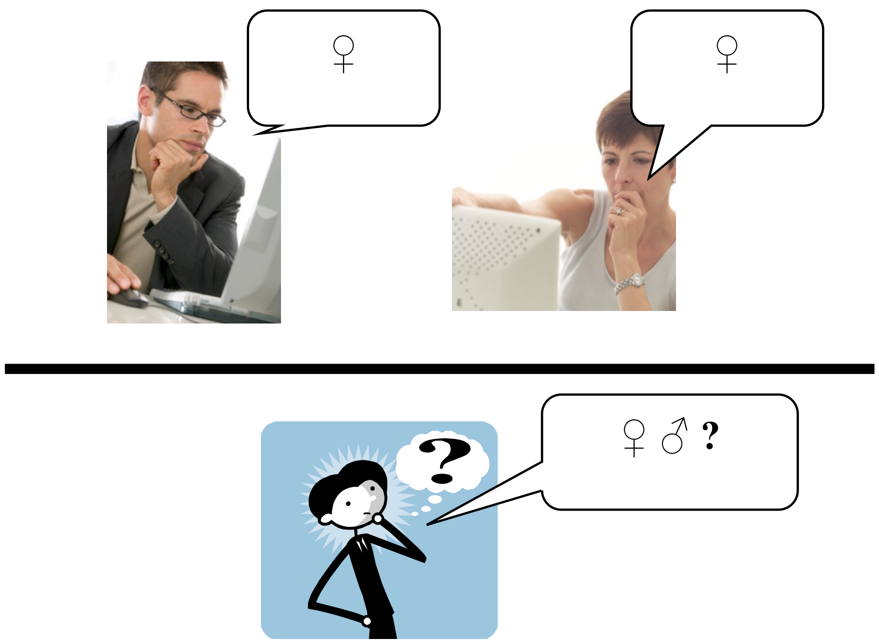
Alan Turing (1912-54)

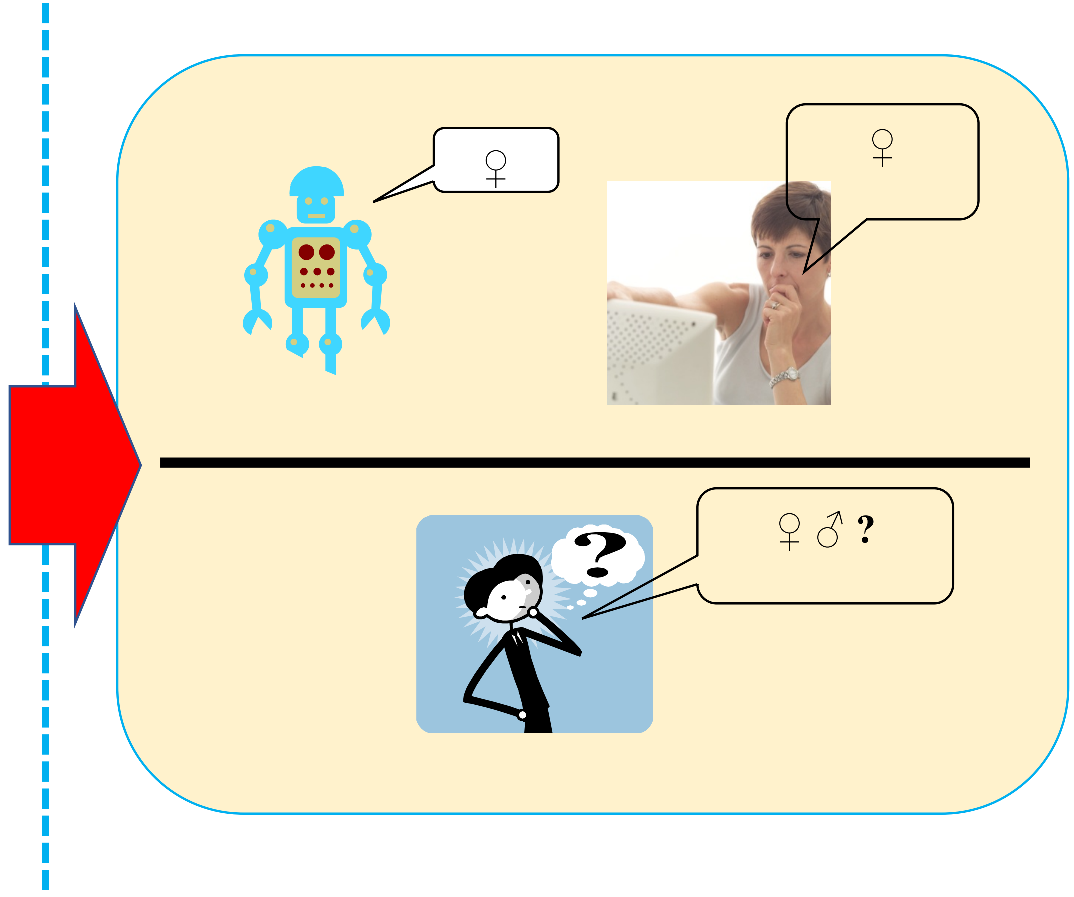


- English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist
- Breaking German cyphers during World War 2 at Bletchley Park
- Founder of the Theoretical Computer Science
  - Principles of computation – Turing machine
  - The Turing Test for defining a standard for a machine to be called "intelligent"

The Turing Test (TT)

- Based on the Imitation Game





The Turing Test - Summary

- Testing intelligence by answering the question "Can machines communicate in natural language in a manner indistinguishable from that of a human being?"
- Arguments against TT and some replies from Turing (see next slides)
- **Reference**:

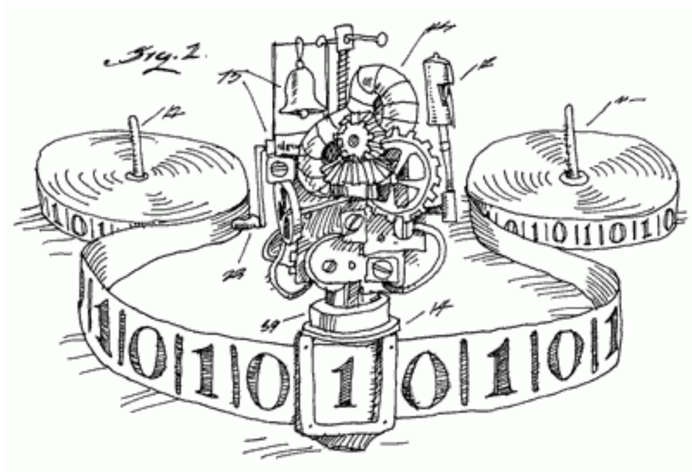
  Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman, "Turing Test: 50 Years later", Minds and Machines 10:463-518, Kluwer, 2000.

Some arguments against TT

- **‘Heads in the sand’ objection**: Thinking machines are not good because they would share human abilities (e.g. thinking).
- **Mathematical objections**: E.g. Gödel’s Theorem (However, maybe intelligent machines can make mistakes)
- **Arguments from consciousness**: Machines should be aware about themselves. Extreme point of view ‘Solipsism’: The only way to really know whether a machine (or man) is thinking or not is to be that machine (or man). Also known as ‘other minds problem’.
- **Arguments from various disabilities**: ‘machines can never do X’ where X is something like ‘have sense of humor’.
- **Lady Lovelace’s objection**: A machine cannot originate anything.

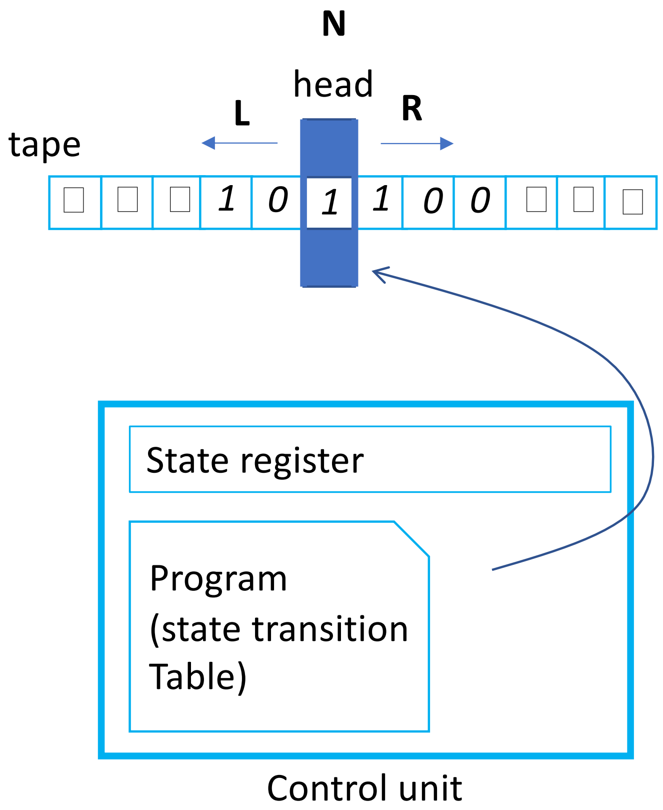
The Turing Machine

- Mathematical model of computation
- Can be used to implement arbitrary computable functions
- Used, e.g., to show that a programming language is as equally expressive as a Turing machine



The Turing Machine

- Consists of:
  - An infinite **tape** divided into cells. Each cell contains a symbol from an **alphabet** or maybe empty (indicated using the **blank** symbol □).
  - A **head** that can read and write symbols on the tape. The head can be **moved left** or **right**.
  - A **state register** storing the current state of the Turing machine.
  - A finite **table** of instructions, where we have for each state an instruction that:
    - Erases or writes symbols
    - Moves the head left (**L**) or right (**R**), or stay at the same place (**N**)
    - Gives the next state of the Turing machine



Definition of a Turing machine

- A Turing machine (TM) is a 7-tuple $(S,\Sigma,\Gamma,s_0,\square,s_e,\Delta)$ *with:*
  - S is a finite set of states
  - Σ is a finite set of input symbols
  - $\Gamma \supset \Sigma$ is a set of possible symbols on the tape
  - $s_0 \in S$ is the start state
  - $\square = \Gamma \setminus \Sigma$ is the blank symbol
  - $s_e \in S$ is the end state, and
  - $\Delta \subseteq S \times \Gamma \times \Gamma \times \{\mathbf{L},\mathbf{R},\mathbf{N}\} \times S$ is the transition table (i.e., the program)

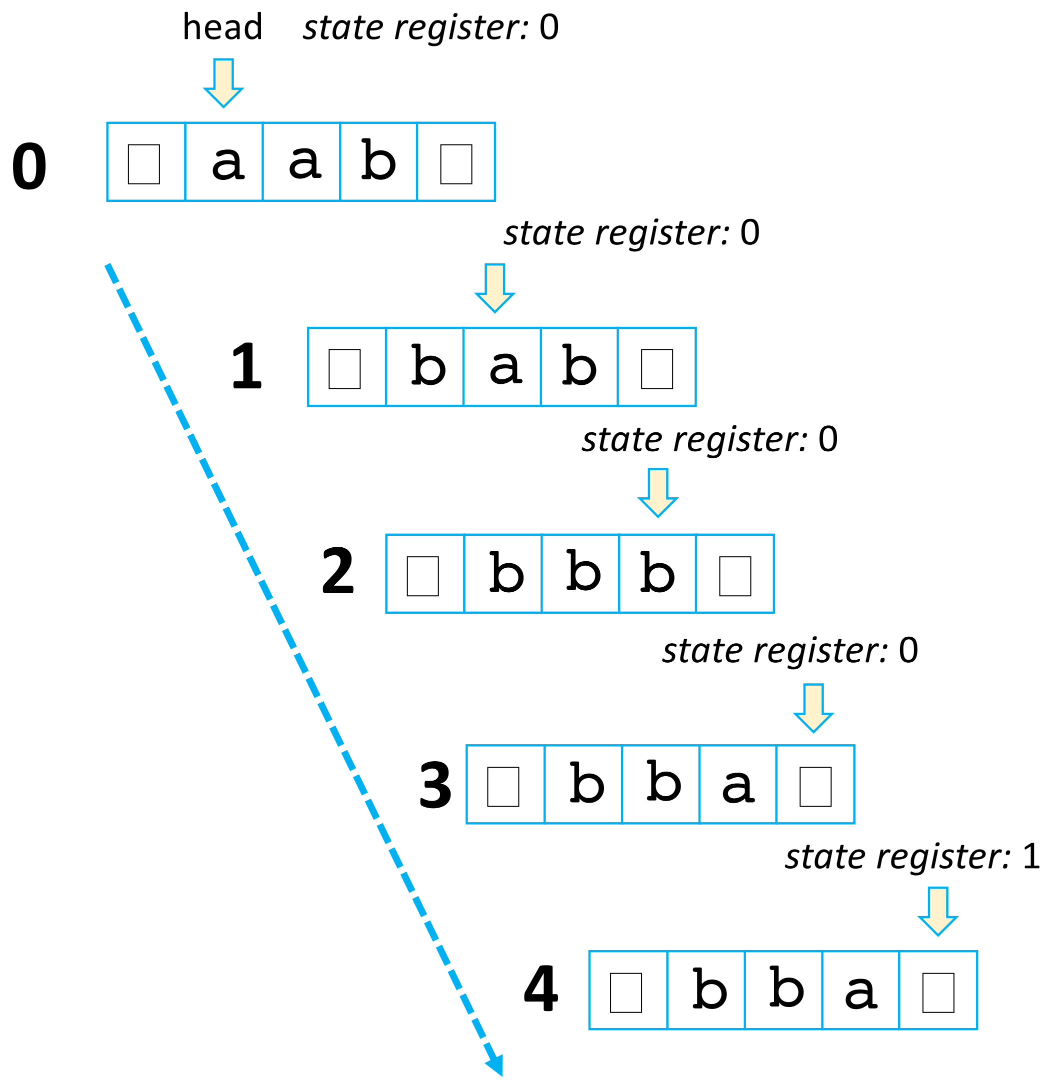
The current state of the TM and the symbol on the tape

The symbol to be written, the action to be carried out, and the next state of the TM

A brief example

- Consider the following TM:
  - S = {0,1}
  - Σ = {a,b}
  - Γ = {a,b, □ }
  - $s_0 = 0$
  - □
  - $s_e = 1$
  - Δ :

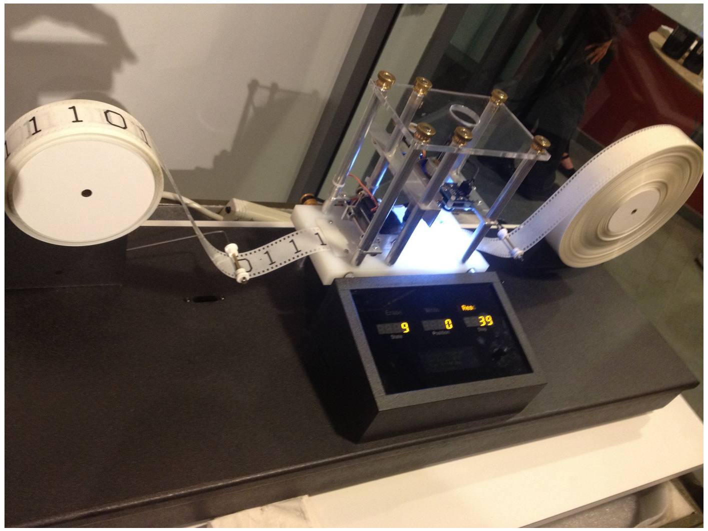
0	a	b	**R**	0
0	b	a	**R**	0
0	□	□	**N**	1



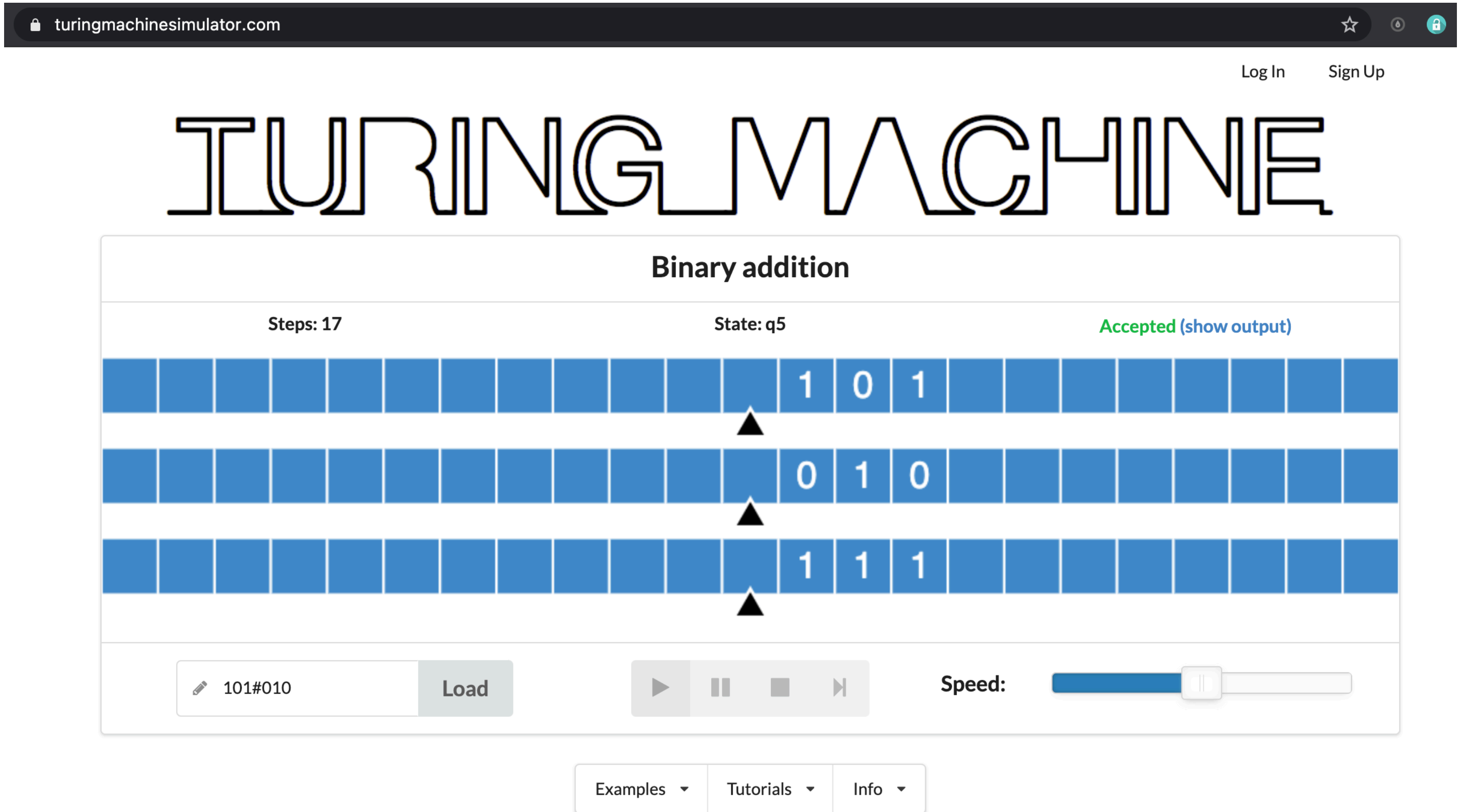
Programming a TM

- Comprises
  - Specifying the input alphabet
  - Defining how things (like numbers) are coded on the tape:
    - Numbers can be coded as binary numbers or a sequence of 1 maybe with some symbol for identifying the begin and end of a number
  - Write the state transition table

- There are “real” TMs
- There are a lot of TM simulators available



TM Simulator



TM capabilities

- We can write programs for arbitrary mathematical functions like plus, minus, etc.
  - Have a look in the internet! There are many examples available including tools for simulating TMs
- There is a correspondence between TM and calculations done by hand:
  - Piece of paper $\rightarrow$ tape
  - Pencil $\rightarrow$ head
  - Calculation is done following an algorithm (in both cases)
  - The result can only be obtained from the tape (or paper) after the completion of the algorithm (i.e., when the TM is in its final state)

Church – Turing Thesis

A function on the natural numbers is computable by a human being following an algorithm, ignoring resource limitations, if and only if it is computable by a Turing machine.

- Every computable function can be computed using Turing machines.

- Does not say anything about efficiency of computation!

TM summary

- TMs implement the mathematical concept of computation
- TMs characterize what is algorithmically computable
- TMs are a general concept of computation
- Every programming language that is Turing complete is as powerful than a TM
- There are different variations of TMs available:
  - Non-deterministic TMs
  - TMs comprising more than one tape
  - TMs distinguishing read from write tapes

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.