Church–Turing Thesis

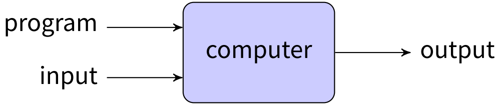
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2017

What is a computer?



A computer is a machine that manipulates data according to a list of instructions

What is a computer?

```
def hello(name):
    print(”Hello ”, name)
```

world

Hello world

Google Now

Will it rain today?

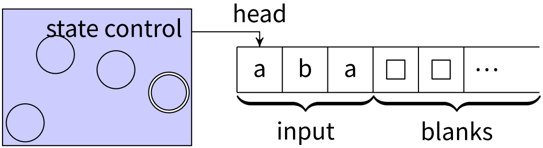
No, rain is not expected today at Ma Liu Shui

Machine Learning algorithm

Environment

self driving

Turing machines



Can both read from and write to the tape

Head can move both left and right

Unlimited tape space

Has two special states accept and reject

Example

$$L_1 = \{ w\#w \mid w \in \{\mathtt{a}, \mathtt{b}\}^* \}$$

Strategy:

Read and remember the first symbol	abbaa#abbaa
Cross it off	xbbaa#abbaa
Read the first symbol past #	xbbaa#abbaa
If they don't match, reject	
If they do, cross it off	xbbaa#xbbaa

Example

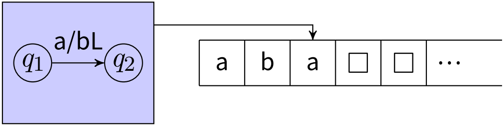
$$L_1 = \{w\#w \mid w \in \{\mathsf{a}, \mathsf{b}\}^*\}$$

Strategy:

Look for and remember the first uncrossed symbol	$\mathsf{x\underline{b}baa\#xbbaa}$
Cross it off	$\mathsf{x\underline{x}baa\#xbbaa}$
Read the first symbol past #	$\mathsf{xxbaa\#x\underline{b}baa}$
If they do, cross it off, else reject	$\mathsf{xxbaa\#x\underline{x}baa}$
At the end, there should be only x's	$\mathsf{xxxxx\#xxxx\underline{x}}$
if so, accept; otherwise reject	

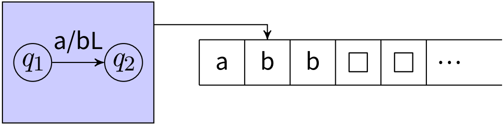
How Turing machines operate

current state: q_1



Replace a with b, and move head left

new state: q_2



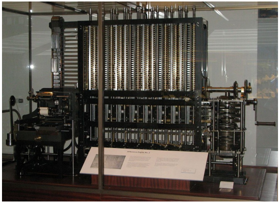
Computing devices: from practice to theory

Brief history of computing devices

Antikythera Mechanism (~100BC)

Its reproduction

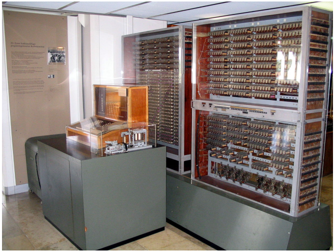
Abacus (Sumer 2700-2300BC, China 1200)



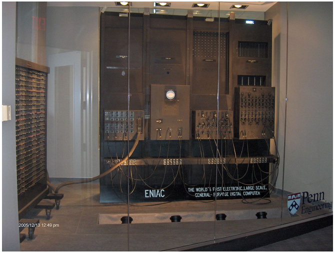
Babbage Difference engine (1840s)

Photo source: Wikipedia

Brief history of computing devices: programmable devices



Z3 (Germany, 1941)



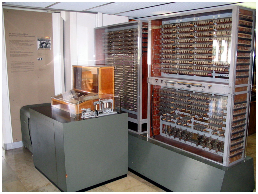
ENIAC (Pennsylvania, US, 1945)

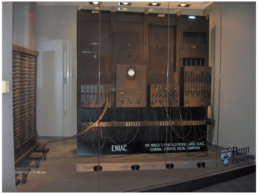
Personal computers (since 1970s)

Mobile phones

Photo source: Wikipedia

Computation is universal



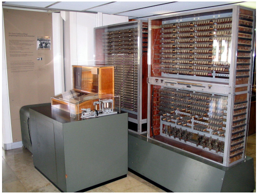


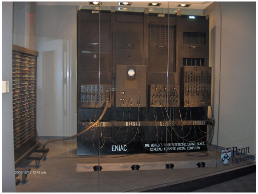
Turing machine

In principle, all computers have the same problem solving ability

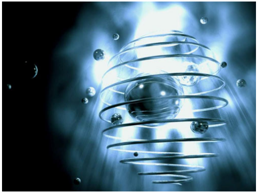
If an algorithm can be implemented on any realistic computer, then it can be implemented on a Turing machine

Church–Turing Thesis

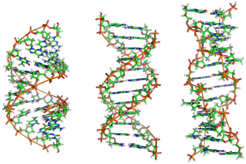




Turing machine

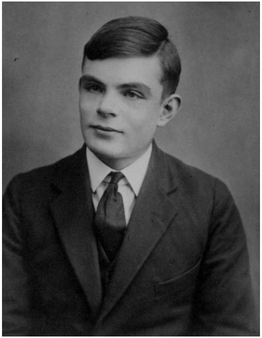


Quantum computing



DNA computing

Alan Turing



Alan Turing aged 16
(1912-1954)

Invented the Turing Test to tell apart humans from computers

Broke German encryption machines during World War II

Turing Award is the “Nobel prize of Computer Science”

Turing’s motivation: Understand the limitations of human computation by studying his “automatic machines”

Hilbert's Entscheidungsproblem, 1928 reformulation

David Hilbert

Entscheidungsproblem (Decision Problem)

"Write a program" to solve the following task:

Input:	mathematical statement (in first-order logic)
Output:	whether the statement is true

In fact, he didn't ask to "write a program", but to "design a procedure"

Examples of statements expressible in first-order logic:

Fermat's last theorem:

$$x^n + y^n = z^n$$

has no integer solution for integer $n \geqslant 3$

Twin prime conjecture:

There are infinitely many pairs of primes of the form p and $p + 2$

Undecidability

Entscheidungsproblem (Decision Problem)

Design a procedure to solve the following task:

Input:	mathematical statement (in first-order logic)
Output:	whether the statement is true

Church (1935-1936) and Turing (1936-1937) independently showed the procedure that Entscheidungsproblem asks for cannot exist!

Definitions of procedure/algorithm:
λ-calculus (Church) and automatic machine (Turing)

Church–Turing Thesis

Church–Turing Thesis

Intuitive notion of algorithms coincides with
those implementable on Turing machines

Supporting arguments:

1. Turing machine is intuitive
2. Many independent definitions of “algorithms” turn out to be equivalent

References:

Alan Turing, “On Computable Numbers, with an Application to the Entscheidungsproblem”, 1937

Alonzo Church, “A Note on the Entscheidungsproblem”, 1936

Formal definition of Turing machine

A Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, where

- Q is a finite set of states
- Σ is the input alphabet, not containing the blank symbol $\square$
- Γ is the tape alphabet ($\Sigma \subseteq \Gamma$) including $\square$
- $q_0 \in Q$ is the initial state
- $q_{\text{acc}}, q_{\text{rej}} \in Q$ are the accepting and rejecting states
- δ is the transition function

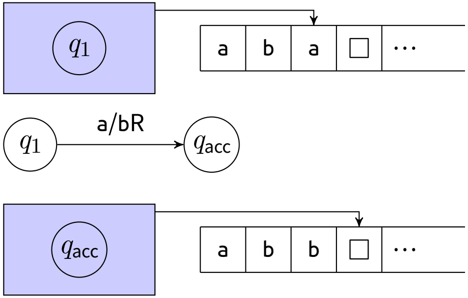
$$\delta : (Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}\}$$

Turing machines are deterministic

Configurations

A configuration consists of current state, head position, and tape contents

Configuration
(abbreviation)



ab q_1 a

abb q_{acc}

Configurations

The start configuration of the TM on input w is $q_0\ w$

We say a configuration C yields C' if the TM can go from C to C' in one step

Example: ab q_1 a yields abb q_{acc}

An accepting configuration is one that contains q_{acc}
A rejecting configuration is one that contains q_{rej}

The language of a Turing machine

A Turing machine M accepts x if there is a sequence of configurations $C_0, C_1, \ldots, C_k$ where

C_0 is starting $\qquad$ C_i yields C_{i+1} $\qquad$ C_k is accepting

The language recognized by M is the set of all strings that M accepts