Programming Language Principles

Programming Language Theory

Notice

- **English**: All Lectures, Assignments, Exams
- **Korean allowed**: Anything written by you.
- There is no disadvantage when you can understand English only.
- Encourage you to practice English, but you can use Korean when it is difficult.
- I’ll give you Korean feedback if you used Korean in individual assignments and questions.

A Few Samples for Questions

- “I don’t understand XXX, can you elaborate?”
  - XXX is the topic you have a hard time to understand.
- “What do you mean by XXXX, in lecture N, slide M?”
  - XXXX is something I said, during lecture N and slide M.
  - Or something I wrote in the slide, or other materials like explantation of assignments.
- “I’d like to know more about XX.”
  - XX would be a topic such as assignment deadline, exam policy, etc.
- Keep it simple, don’t need to feel pressure to put all the details. I’ll figure it out.

Topics

- **What is a Computer?**
- **Turing Machine**
- How to implement a PL?
  - Compiler & Interpreter

What is a Computer?

- What is a computer in PL’s perspective?
- Programming languages eventually run on computers.
- To design a programing language, or develop a program with it → it is necessary to understand how a computer works.

What is a Computer?

- When you hear this question, there might be various images of computers on your mind.
- In this week's lectures, we will explore this question more theoretically.
- After the lectures, you will have general, universal and more theoretical view of a computer.

What should we consider?

- When we run a PL on a computer, what should we consider?
- In a PL’s eyes, a computer is providing something like these,
    - Data types
    - Operators
    - Control of Execution

What should we consider?

- Control of Data
- Memory Management
- Input and Output (I/O)

Data Types

- When a computer is doing a computation, the computation is often performed on data.

- There exist various data types, and ***applicable computations are dependent on data types***.

- Data types should be considered to ***verify the correctness of computation*** and also to ***choose a correct computation***.

Operators

- It looks like a computer can handle a complex computation easily.
- However, it combines various basic operations to deal with such complex computations internally.
- How can a computer process multiplication and division?
    - e.g.) using shifter and adder or subtractor.

Control of Execution

- A computer should control its execution of operations.
- e.g.) Executing some operations repeatedly, or executing only a part of operations.
- To obtain a desired outcome, we need to execute operations based on our intention.

Control of Data

- In a computer, CPU eventually processes data which are being computed.
- However, this data do not exist in CPU at first.
- Hence it is necessary to control the flow of data inside a computer.

Memory Management

- When a computer executes a program, usually the program is loaded to memory.
- What if a program itself is larger than available memory?
- Appropriate memory management is necessary to load and remove data from memory.

Input and Output

- When a user is using a computer,
    - the computer gets input from the user,
    - and it provides output to the user.
- Usually I/O takes a huge amount of the processing time, hence a computer should handle it effectively.

So What?

- We know what should be considered, but each programming language will handle these matters differently.
    - e.g.) languages w/ unconditional branch (goto) vs. languages w/o unconditional branch
- Can we define a computer in more general, theoretical ways?

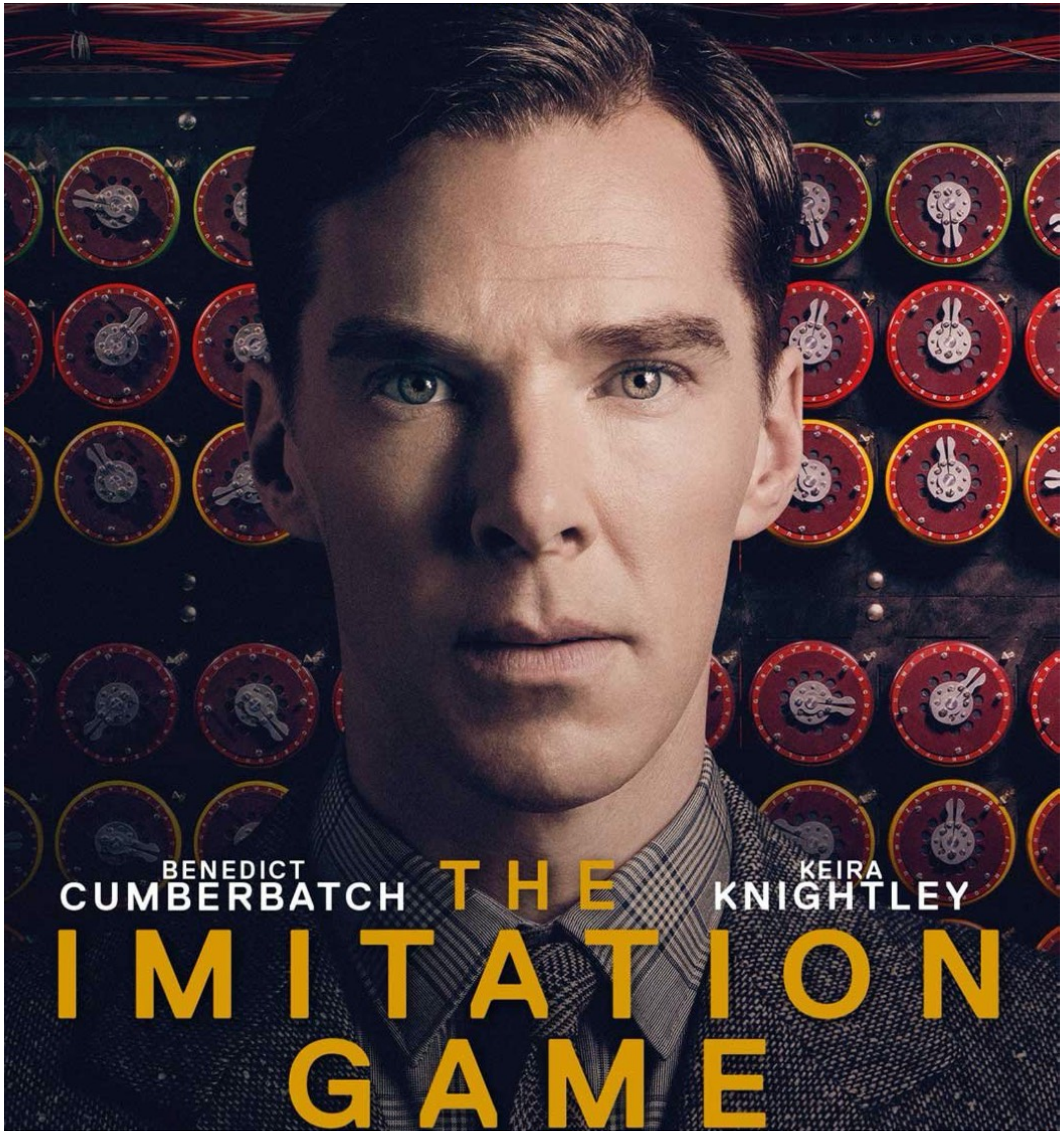
Turing Machine

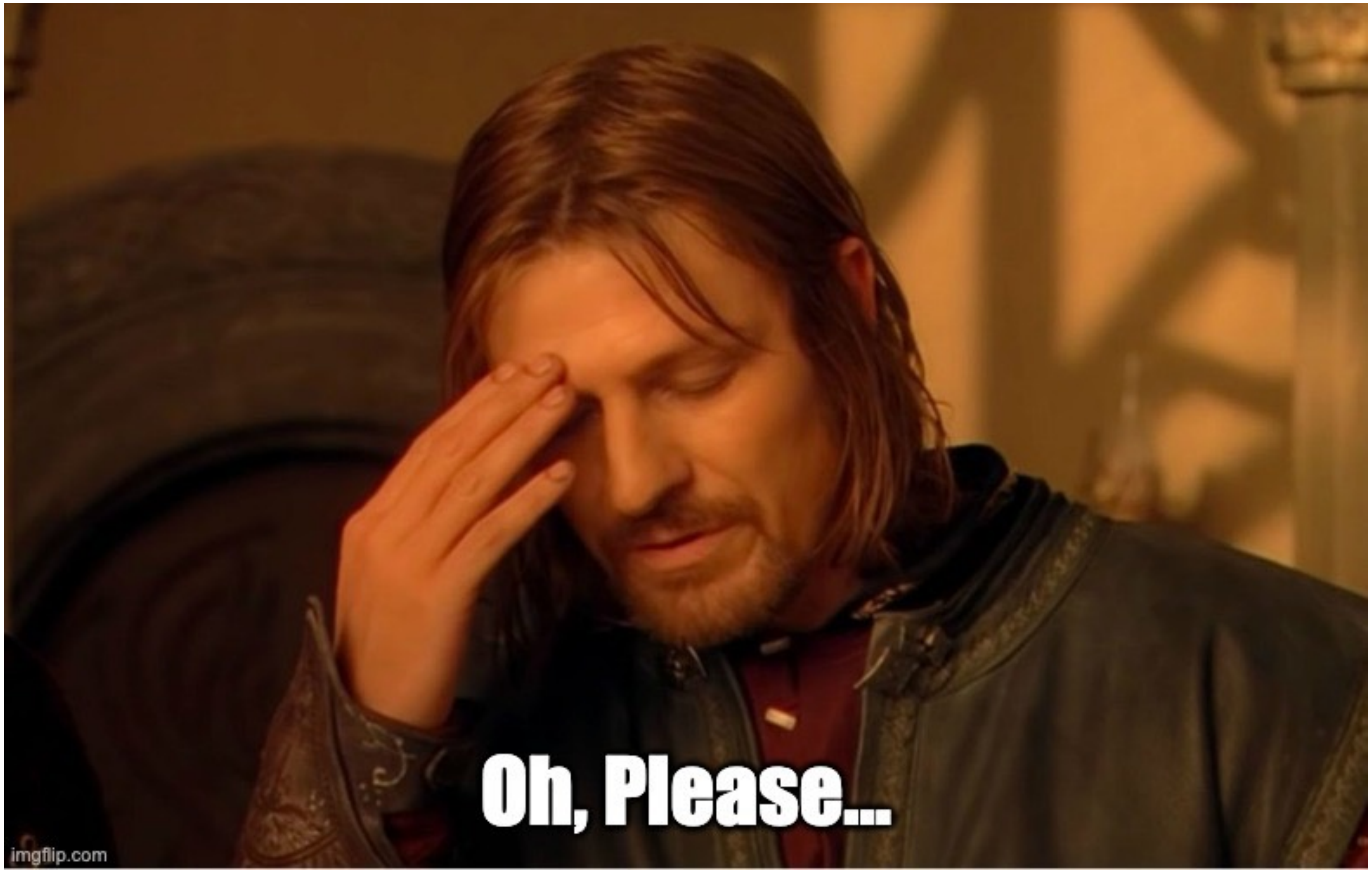
- First introduced in 1936 by Alan Turing.
- Originally it was called “a-machine”, which means automatic machine.
- It was a theoretical, imaginary machine invented to prove properties of computation in general.
- Later it became a foundation of modern computers.

Do you know Alan Turing?

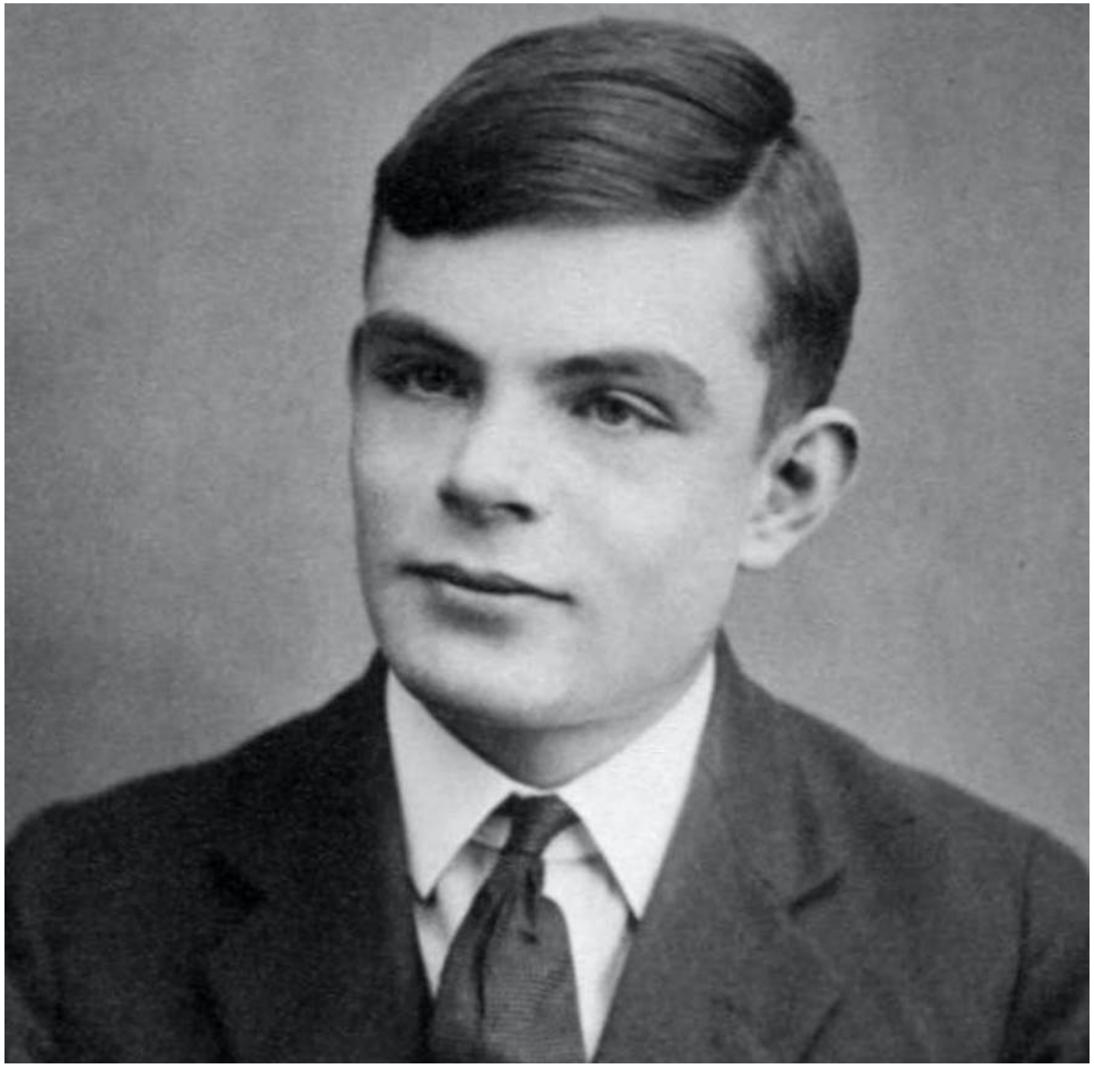
Yes, I Do!

This Guy!

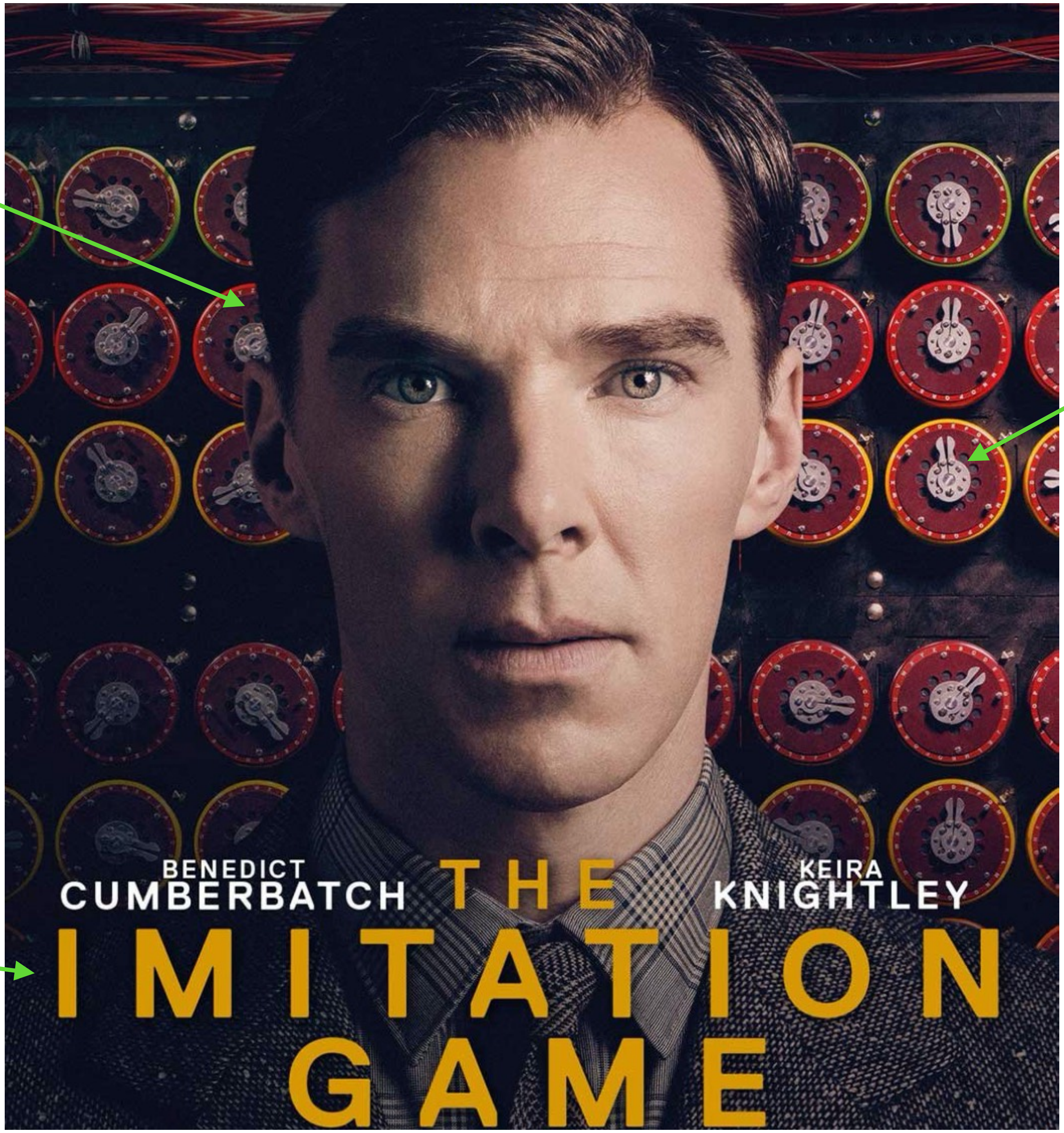



Oh, Please...
imgflip.com

No! This Guy!



Disturbing Points

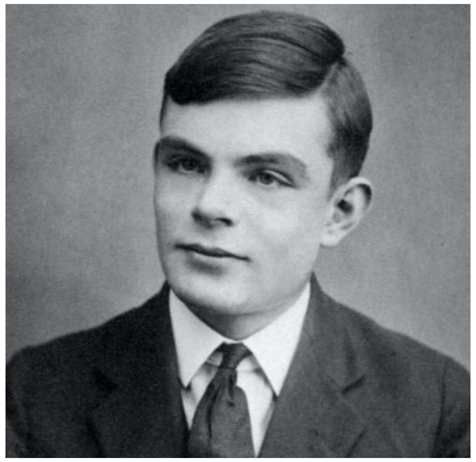


He is not Turing!

This machine is not Enigma!

The movie is not related to the imitation game!

Alan Turing
(1912~1954)



- British computer scientist, logician, cryptanalyst.
- **Imitation Game**: a.k.a Turing Test. First introduced by his paper “*Computing Machinery and Intelligence*” in 1950.
- It is about how to verify whether machines can think (or imitate human) or not.
- **Halting Problem**: Proved the existence of *undecidable* problem.
- He built a foundation of theoretical computer science.

Turing Machine (cont’d)

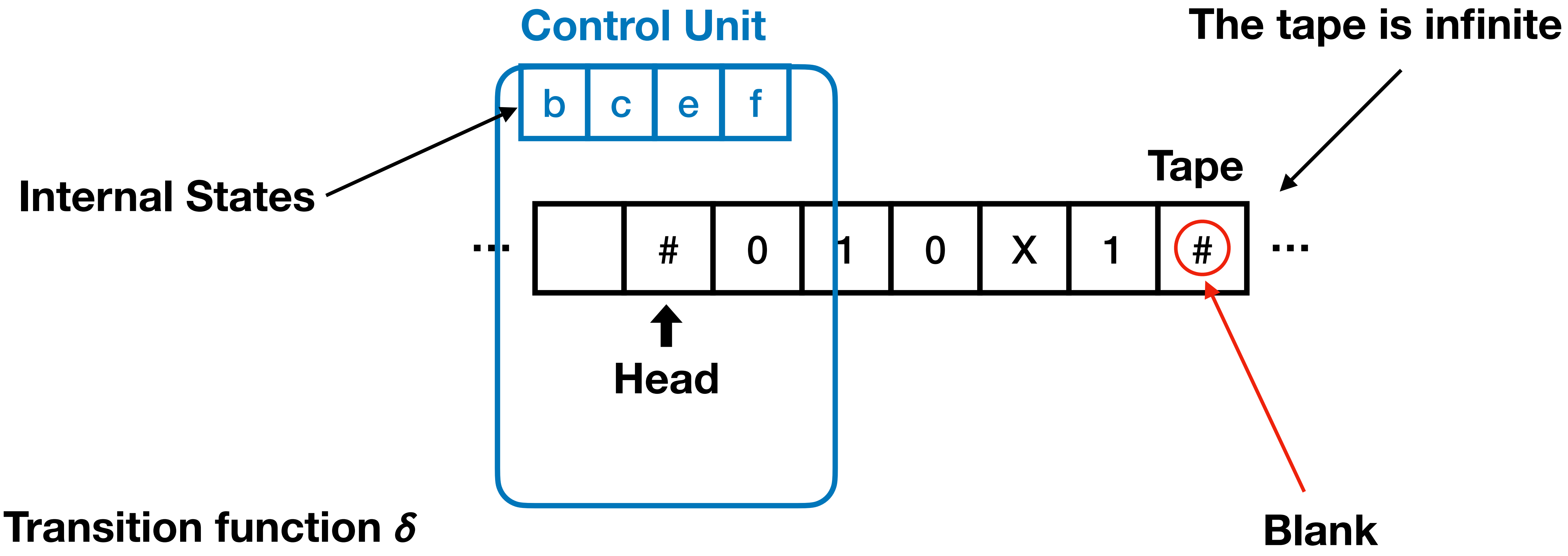
- Turing machine consists of a control unit and an infinite tape.

- A **tape** is divided into cells, and each **cell** contains one symbol.

- **Head** points to the current cell, and it can read or write a symbol to the cell.

- **Control unit** controls the move of the head, left or right, and performs a certain operation based on the current symbol.

Turing Machine (cont’d)



Transition function δ

current state	symbol	operations	final state
b	#	P0, R	c
c	#	R	e
e	#	P1, R	f
f	#	R	b

Turing Completeness

- So far, it is known that all ***computational problems*** can be solved by a Turing machine.
  - e.g.) Anything can be done by computers can also be done by a Turing machine.
- A system is ***Turing complete***, if it can be used to simulate a Turing machine.
- A Turing complete system has equivalent ability of computation as a Turing machine.

Summary

- Computer in PL’s perspective
- Turing Machine and Turing Completeness