Turing Machine (TM)

Founders of the ICT Era

KENTECH Korea Institute of Energy Technology

AI

Alan Turing (1912 – 1954), *the Father of Computer Science*, was an English mathematician,
computer scientist, logician, cryptanalyst.
Turing was highly influential in the development of theoretical computer science, his Ph.D dissertation provides formalization of the concepts of algorithm and computation with the Turing machine.

Claude Shannon (1916 – 2001), *the Father of Information & Communications*, was an American mathematician, electrical engineer.
His master thesis on switch theory demonstrated that electrical applications of Boolean algebra could construct any logical numerical relationship. Another notable Shannon’s work is information theory.

Early in 1943, Shannon came into contact with Alan Turing. Turing had been posted to Washington to share with the U.S. Navy's cryptanalytic service the his code breaking methods. Shannon and Turing met at teatime in the cafeteria. Turing showed Shannon his 1936 paper on the "Universal Turing machine". This impressed Shannon, as many of its ideas complemented his own.

1900 – At the turn of the millennium

KENTECH Korea Institute of Energy Technology AI

- At Paris, France

- At ICM (Intn'l Cong. Of Math.),
- David Hilbert announced his famous list of 23 (Or 24) unsolved mathematical problems
- An important theme is not simply proving more theorems true, but achieving *automation of theorem-proving* itself

David Hilbert (1862 – 1943), was a German mathematician and one of the most influential mathematicians of the 19th and early 20th centuries.

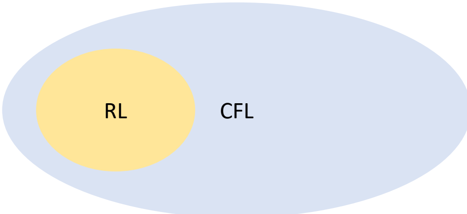
Hilbert's Vision

KENTECH Korea Institute of Energy Technology AI

- **1917:** Hilbert distinguishes first-order logic from other sorts of logical systems and presents it in a class at the University of Göttingen
- **1928:** Hilbert poses the *Entscheidungsproblem ("decision problem")*, asking whether there is a mechanical procedure by which mathematical theorems can be automatically proven or disproven, setting up a challenge to define what a "mechanical procedure" means and how one might build such a procedure for automated theorem proving

The Problem

AI



- Finite automata accept precisely the RL(Regular Languages)
- We may need unbounded memory to recognize context-free languages
  - e.g. { $a^n b^n$ | n ∈ ℕ } requires unbounded counting
- How do we model a computing device that has unbounded memory?

Hilvert's Vision & Reality

AI

- Hilbert's agenda
  - Inspires some of the most impactful theoretical work in mathematical history, human history
  - Brings us heroes like Alonso Church, Alan Turing and Kurt Gödel!
- However, Turing, Church, Gödel and others proved automation can never prove all mathematical facts true!
- These incredible results came about because of Hilbert, but demolished his vision of automated knowledge creation

Turing Discovered...

KENTECH
Korea Institute of Energy Technology
AI

- Key Idea: Even if you need huge (unlimited) amounts of scratch space to perform a calculation, at each point in the calculation you only need access to a small amount of that scratch space

```
   3398644231057743621096537623875319175812
+  4845873986442310880621051053762990760093
-------------------------------------------
                                          5
```


Turing Machine (TM)

Turing Machine

KENTECH Korea Institute of Energy Technology AI

- To provide his machines extra memory, Turing gave his machines access to an infinite tape subdivided into a number of tape cells
- A Turing machine can only see one tape cell at a time, the one pointed at by the tape head

- TM can
  - Read the cell under the tape head
  - Change symbol written under the tape head
  - Move its tape head to the left or to the right

…		d	e	c	r	e			…

Turing Machine

KENTECH Korea Institute of Energy Technology AI

- TM can
  - Read the cell under the tape head
  - Change symbol written under the tape head
  - Move its tape head to the left or to the right

…		i	e	c	r	e			…

…		i	n	c	r	e			…

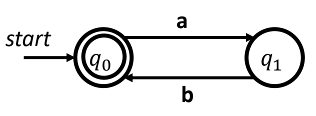
…		i	n	c	r	e			…

TM Commands

- Six commands:
  - Move *direction* : Move *left* or *right* by one
  - Write *symbol* : Write *symbol* under the head
  - Goto *label* : Move to the *label*
  - Return *result* : Terminate with return *result* (Either True Or False)
  - If *symbol command* : If the symbol under the head is *symbol* , then perform the *command*
  - If Not *symbol command* : If the symbol under the head is NOT *symbol* , then perform the *command*
- Despite their simplicity, TMs are surprisingly ***powerful***

TM -Example

- Write a TM whose language is equivalent to the regex (ab)*

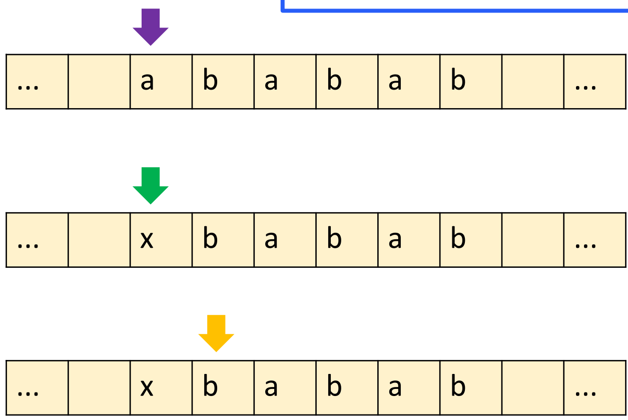



```
Start:
   If Blank Return True
   If 'b' Return False
   Write 'x'
   Move Right
   If Not 'b' Return False
   Write 'x'
   Move Right
   Goto Start
```

...		a	b	a	b	a	b		...

TM -Example

```
Start:
    If Blank Return True
    If 'b' Return False
    Write 'x'
    Move Right
    If Not 'b' Return False
    Write 'x'
    Move Right
    Goto Start
```



TM for NRL(NonRegular Language)

- The language L = { $a^n b^n$ | n ∈ ℕ } is a canonical example of an NRL
- It's not possible to check if a string is in this language given only finite memory
- Turing machines, however, are powerful enough to test NRL

Use Recursion

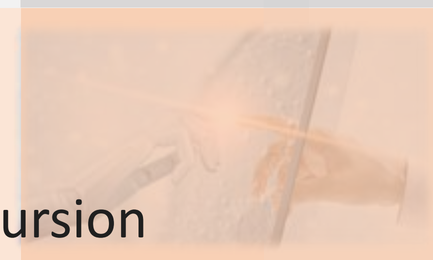
Addition

- Can a TM add two natural numbers?
- How to represent two natural numbers?
  - ➔ Number of stones

- Example: 2 + 3
  - Input

  - Output

Write a program that adds !!



Turing Machine - Recursion

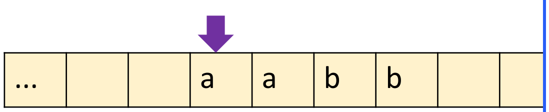
Recursion

- We can process our string using a recursive approach
  - The string ε is in *L*
  - The string awb is in *L* if and only if w is in *L*
  - Any string starting with b is not in *L*
  - Any string ending with a is not in *L*

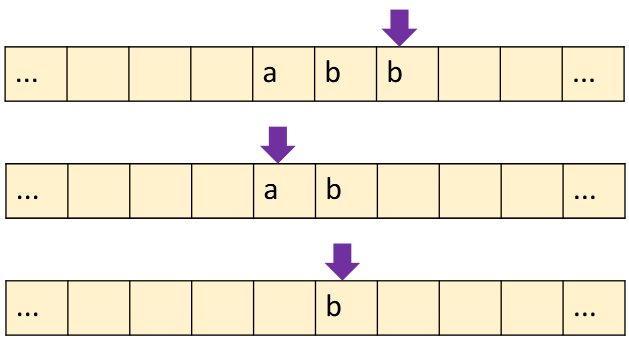
- Now, how to develop
  a TM that implements this

Start:
    If the first character is a,
        Erase a (Write blank)
    Move to the end of the string
    If the last is b, Write blank
    Move to the start of the string
    Repeat

Simple Case Simulation



Start:
    If the first character is a,
        Erase a (Write blank)
    Move to the end of the string
    If the last is b, Write blank
    Move to the start of the string
    Repeat



TM for NRL

```
Start:
    If Blank Return True
    If 'b' Return False
    Write Blank

ZipRight:
    Move Right
    If Not Blank Goto ZipRight
    Move Left
    If Not 'b' Return False
    Write Blank

ZipLeft:
    Move Left
    If Not Blank Goto ZipLeft
    Move Right
    Goto Start
```

…			a	a	b	b			…

A More Complicated Problem

- How about

  L = { w ∈ {a, b}* | $|w_a|$ = $|w_b|$ }

  where $|w_a|$, $|w_b|$ are the numbers of a's and b's in w }
- L is not regular (Prove it!)
- L is context-free (How about its CFG?)

Try to design a TM yourself
You surely can sketch (design) a correct one

Hint: A TM can use (write) separate *tape alphabet* in addition to Σ
For example, if Σ = {a,b}, then a TM can use x

Design → Coding requires training

TM for L = { w ∈ {a, b}* | $|w_a|$ = $|w_b|$ }

AI

...		a	a	b	b	a	b		...

...			a	b	b	a	b		...

...			a		b	a	b		...

Should distinguish input blanks from intentional blanks
➔ Use other special characters as tape alphabet
➔ Use x

TM for L = { w ∈ {a, b}* | $|w_a|$ = $|w_b|$ }

AI

```
Start:
   If 'a' Goto FoundA
   If ‘b' Goto FoundB
   If Blank Return True
   Move Right
   Goto Start

FoundA:
   Write 'x'
LoopA:
   Move Right
   If 'a' Goto LoopA
   If 'x' Goto LoopA
   If Blank Return False
   Write 'x'
  Goto GoHome

GoHome:
   Move Left
   If Not Blank Goto GoHome
   Move Right
   Goto Start

FoundB:
   Write 'x'
LoopB:
   Move Right
   If 'b' Goto LoopB
   If 'x' Goto LoopB
   If Blank Return False
   Write 'x'
   Goto GoHome
```

TM for L = { w ∈ {a, b}* | $|w_a|$ = $|w_b|$ }

KENTECH Korea Institute of Energy Technology AI

…		a	a	b	b	b	a		…

```
Start:
   If 'a' Goto FoundA
   If 'b' Goto FoundB
   If Blank Return True
   Move Right
   Goto Start

FoundA:
   Write 'x'
LoopA:
   Move Right
   If 'a' Goto LoopA
   If 'x' Goto LoopA
   If Blank Return False
   Write 'x'
  Goto GoHome

GoHome:
   Move Left
   If Not Blank Goto GoHome
   Move Right
   Goto Start

FoundB:
   Write 'x'
LoopB:
   Move Right
   If 'b' Goto LoopB
   If 'x' Goto LoopB
   If Blank Return False
   Write 'x'
   Goto GoHome
```


Another Idea

KENTECH Korea Institute of Energy Technology AI

- If all ‘a’s appear before ‘b’, then it becomes a much easier problem, $L = \{ a^n b^n \mid n \in \mathbb{N} \}$
- Can you sort the characters in input strings?

…			b	a	a	b			…

…			b	a->b	a	b			…

…			b->a	b	a	b			…

…			b->a	b	a->b	b			…

Key Idea: Abstraction

KENTECH Korea Institute of Energy Technology AI

- Once we have a Turing Machine that does one task, we can use it much like a helper function call for more complex Turing Machines
- This idea will prove key to the power and versatility of TMs