INT201 Decision, Computation and Language

Lecture 10 – Church-Turing Thesis and Limits of Computation

Dr Yushi Li and Dr. Chunchuan Lyu

Xi'an Jiaotong-Liverpool University
西交利物浦大學

1

Recap

- Turing Machine
- Turing-recognizable (halts on accept) and Turing-decidable (always halts) languages
- Multi-tape TM and Nondeterministic TM

Today

- Cantor's Diagonalization Method
- Church-Turing Thesis and Universal Turing Machine
- Examples of decidable languages
- Existence of undecidable language and non-Turing recognizable language

2

Diagonalization method 对角线法

实际上除了通过数集合中元素的多少来对比集合的大小，还可以通过对两个集合中的元素进行配对来对比规模

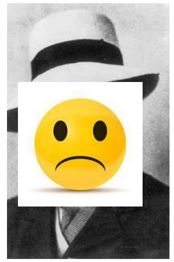
Questions:

用自然数去对应自然数
If we use natural numbers to list natural numbers,
couldn't we construct a new number by flipping the diagonal and get a new number that is not in the natural number set?

Answer:

True, but the number you get is of infinite length (countable length).
All natural numbers are of finite length.
Therefore, you actually constructed a real number.
There is nothing wrong with a real number not in the natural number set.

A Brief History of the Limits of Computation



David Hilbert
1862-1943
German
Mathematician

- The existence of uncountable sets - Georg Cantor 1874
- The diagonal method - Georg Cantor 1891
- Is there a set between real and natural numbers? – David Hilbert 1900
- Prove axioms of arithmetic are consistent – David Hilbert 1900
- **We must know. We shall know – David Hilbert 1930**
- The existence of non-provable & non-disprovable statements
- Consistency of powerful system cannot be proved within itself - Kurt Gödel 1931
- Whether a statement is provable from axioms is not Turing-decidable – Church & Turing 1936
- Is there a set between real and natural numbers is independent of ZFC –Gödel 1940 & Cohen 1963
- **The Church–Turing Thesis**: every effectively calculable function (effectively decidable predicate) is general recursive (Turing computable)– Stephen Kleene 1952

The Church–Turing Thesis

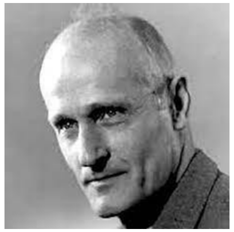
Alonzo Church
1903 - 1995

Algorithm = Turing machine

Intuitive — Formal

Alan Turing
1912–1954

How to prove this? Is it even possible?



Stephen Kleene
1909 -1994

How to disprove this? Is it even possible?

The Church–Turing Thesis 丘奇图灵理论

Existence of non Turing-recognizable languages. 存在非图灵识别语言

命题：每个图灵机都可以用1和0以及一些有限的特殊符号组成的不同的有限字符串进行编码。

Proposition: Each Turing machine can be encoded by a distinct, finite string of 1's and 0's and some finite special symbols.

Encoding：可以将一个图灵机的表示用特定的规则

Proof : encode TM as 7 tuple with special symbols, and encode alphabets in binary. Transitions can be encoded as a sequence of 5 tuples (state,tape,new state, new tape, left or right).

It is of critical importance that each single Turing machine is described in finite length. 至关重要的是，每个图灵机的描述长度是有限的

Corollary: There are countable many Turing machines.

推论：存在可数多的图灵机

If a problem cannot be done on a TM, then no computer can solve it

The Church–Turing Thesis
Existence of non Turing-recognizable languages.

1.可计算性：如果一个函数是countable，那么它可以通过一个图灵机来计算。换句话说，如果一个函数能被任何图灵机计算，那么它就是可计算的
2.等价性：如果一个问题可以被图灵机解决，那么也可以用等价模型解决
3.普遍性：通用计算模型，TM是计算极限，不存在更强计算模型
4.实际计算极限：图灵机计算能力代表了实际可计算性能，任何物理过程都可以被TM模拟。

推论

Corollary: There are countable many Turing machines.

命题：

Proposition: There are uncountable many languages.

$A \subseteq \Sigma^*$

Proof:
$$
\begin{aligned}
\Sigma^* &= \{\ \varepsilon,\ 0,\ 1,\ 00,\ 01,\ 10,\ 11,\ 000,\ 001,\ \cdots\ \}\ ; \\
A &= \{\ \ 0,\ \ 00,\ 01,\ \ 000,\ 001,\ \cdots\ \}\ ; \\
\chi_A &= \ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ \cdots\ .
\end{aligned}
$$

有1或空是0

There is a bijection between languages and countable binary sequences that is uncountable by the diagonalization method.

双射

Corollary There exists non Turing-recognizable languages.

Is there another languages between all languages and Turing-recognizable languages?
This is equivalent to the existence of a set number natural numbers and real numbers. So, it is independent of ZFC

TM description：状态集合(Q)，输入字母表(Σ)，带字母表(Γ)，转移函数(δ)
起始状态(q_0)，接受和拒绝(q_{accept} / q_{reject})

The Church–Turing Thesis
Universal Turing Machine 在转换的时候，可以按照特定方式排列

Encoding

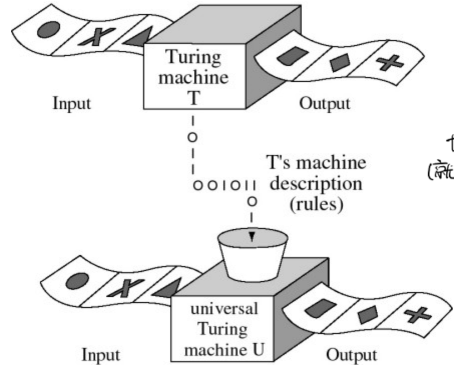
(在q_0读a 后写入a然后去到q_1)

$\langle M\rangle =$ " $Q = q_0, q_1, q_2, q_a, q_r$; $\Sigma = a, b$; $\Gamma = a, b$; q_0 ; $F = q_a$; $\delta = (q_0, a, a, R, q_1)$

<>代表encoding，<D, W>可以代表encoding后的DFA，例如：#Q#Σ#δ#q_0#F#w#

Theorem Universal Turing Machine Exists 通用图灵机是否存在？

There exists a TM U that takes a Turing machine description and input tape and simulate one step of that given Turing machine on the input tape. 存在一个TM U，它采用图灵机描述和输入磁带，并在输入磁带上模拟给定图灵机的一个步骤。



Input to TM U is in the format <M,w> where
M is the TM description, and
w the (converted) input.

transforming machines into string representation
(就像java的toString()，只不过有自己的变化方式)

The smallest UTM has 2 states and 3 symbols - Alex Smith, a 20-year-old undergraduate from Birmingham 2007

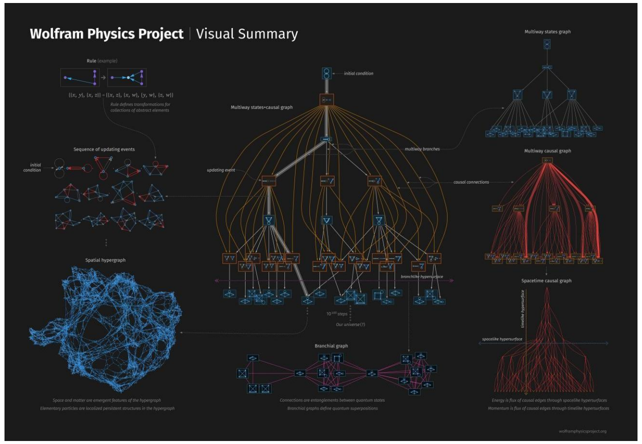
Church-Turing Thesis

To **prove** the thesis, we need to show that the world is Turing computable.

The Minecraft world is simulated on digital computer, and we can build computer inside Minecraft. Hence, in the Minecraft world, computation is equivalent to Turing computable.

Church-Turing Thesis

To **prove** the thesis, we need to show that the world is Turing computable.



Stephen Wolfram has a hyper graph replacement based formalism for a theory of everything.

If a theory like this is true, then the world is Turing computable. (how can we know?)

https://writings.stephenwolfram.com/2020/04/finally-we-may-have-a-path-to-the-fundamental-theory-of-physics-and-its-beautiful/

丘奇·图灵论题

Church-Turing Thesis

To **prove** the thesis, we need to show that the world is Turing equivalent *up to manipulating a sequence of finite symbols.*

Q: but we live in a quantum universe, clearly there are things that cannot be captured by discrete symbols.

A: Turing machine examines a finite sequence of symbols, it cannot represent all mathematical object.
The question is that given your extra power in the physical world, can you do more in terms of recognizing a finite sequence of symbols?

In fact, quantum Turing machine is equivalent to Turing machine.

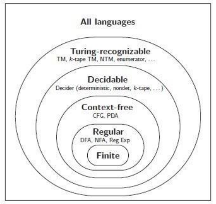
Also, Church-Turing thesis is not about time complexity.

Church-Turing Thesis

反证 我们需要展示没有图灵可识别语言可以被(物理设备识别)

To **disprove** the thesis, we need to show that there is a non Turing-recognizable/decidable language that can be recognized or decided by a physical device.

This process is how we find PDA on top of DFA, and TM on top of PDA.



Can we draw another circle? With possible overlap

If we can find a machine that manipulate the tape in a way that TM cannot simulate in finite time, we can construct a non Turing-recognizable language.

Church-Turing Thesis

图灵的原始观点：图灵通过建立关键限制，说明人类计算可以简化为有限的一组简单机械操作。

Turing's original argument where Turing shows that human computation can be reduced to a finite set of simple mechanical operations by establishing key limitations:

- Finite symbols: Humans can only write/recognize a limited set of distinct symbols
- Limited observation: We can only view a bounded number of squares at once
- Local movement: We can only move our attention within a fixed distance
- Direct manipulation: We must observe a square to modify it
- Deterministic behavior: Actions are determined by current observations and mental state
- Finite mental states: The number of possible mental states is bounded
- Elementary operations: All computation reduces to simple atomic actions (changing mental state, moving attention, or modifying one symbol)

在形式上，如果存在一个TM，对于任何给定输入，都能在有限的步骤内停止并接受或拒绝该输入，那么我们就说这个问题是可判定的

1、有解：对于所有合法输入，TM都能停止，并处于 accept 或 reject.

2. 有限时间内：TM在停止前不会一直 loop.

Decidability

Given a language L whose elements are pairs of the form (B, w), where

- B is some computation model (e, g. DFA, NFA…).
- w is a string over the alphabet Σ.

The pair $(B, w) \in L$ if and only if $w \in L(B)$.

Since the input to computation model B is a string over Σ, we must encode the pair (B, w) as a string.

Acceptance problem for computation model

决策问题

Decision problem: Dose a given model accept/generate a given string w?

Instance $\langle B, w\rangle$ is the encoding of the pair (B, w).

Universe Ω comprises every possible instance:

$$\Omega = \{\langle B, w\rangle \mid B \text{ is a model and } w \text{ is a string}\}$$

Language comprises all "yes" instances

$$L = \{\langle B, w\rangle \mid B \text{ is a model that accept } w\} \subseteq \Omega$$

Acceptance problem for Language L_{DFA}

Decision problem: Dose a given DFA B accept a given string w?

Instance $\langle B, w\rangle$ is the encoding of the pair (B, w).

Universe Ω comprises every possible instance:

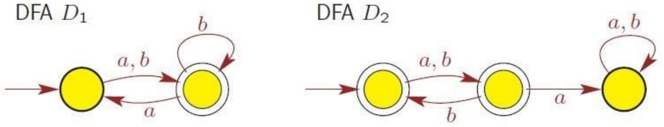
$$\Omega = \{\langle B, w\rangle \mid B \text{ is a DFA and } w \text{ is a string}\}$$

Language comprises all "yes" instances

$$L = \{\langle B, w\rangle \mid B \text{ is a DFA that accept } w\} \subseteq \Omega$$

Acceptance problem for Language L_{DFA}

Example



$\langle D_1, abb\rangle \in A_{\mathsf{DFA}}$ and $\langle D_2, \varepsilon\rangle \in A_{\mathsf{DFA}}$ are YES instances.
$\langle D_1, \varepsilon\rangle \notin A_{\mathsf{DFA}}$ and $\langle D_2, aab\rangle \notin A_{\mathsf{DFA}}$ are NO instances.

encoding $<D, w>$ $\# Q \# \Sigma \# \delta \# q_0 \# F \# w_1 \ldots\ldots w_n \# \underbrace{q_i \ldots\ldots}_{\text{store current state}}$

用TM M来判定

然后可以每读取一个w_i就删掉
把新到达的 state 添加在后面
最后当 w 没有之后 看 state
是否在 accept state.

The Language L_{DFA} is decidable

$L_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accept } w\} \subseteq \Omega$

$\Omega = \{\langle B, w\rangle \mid B \text{ is a DFA and } w \text{ is a string}\}$

如果在 F $\rightarrow$ TM go to q_{accept}
如果不在 F $\rightarrow$ TM go to q_{reject}

To prove L_{DFA} is decidable, we need to construct TM M that decides L_{DFA}.

For M that decides L_{DFA} :

- take $\langle B, w\rangle \in \Omega$ as input
- halt and **accept** if $\langle B, w\rangle \in L_{DFA}$
- halt and **reject** if $\langle B, w\rangle \in L_{DFA}$

The Language L_{DFA} is decidable

Proof

Basic idea:

On input $\langle B, w\rangle \in \Omega$, where

- $B = (\Sigma, Q, \delta, q_0, F)$ is a DFA
- $w = w_1 w_2 \cdots w_n \in \Sigma^*$ is input string to process on B.

1. Check if $\langle B, w\rangle$ is "proper" encoding. If not, reject
2. Simulate B on w based on: encoding 为 #Q#Σ#δ#q_0#F#$w_1 \dots w_n$#

- $q \in Q$, the current state of B
- $i \in \{1, 2, \dots, |w|\}$, the pointer that illustrates the current position in w. 不断更新指针 指向下一个要输入的字符
- q changes in accordance with w_i and the transition function $\delta(q, w_i)$. 不断更新 current state

3. If B ends in $q \in F$, then M accepts; otherwise, reject.

L_{NFA} is decidable because L_{DFA} is decidable

The Language L_{NFA} is decidable

Decision problem: Dose a given NFA B accept a given string w?

$$L_{DFA} = \{\langle B, w\rangle \mid B \text{ is a NFA that accept } w\} \subseteq \Omega$$

$$\Omega = \{\langle B, w\rangle \mid B \text{ is a NFA and } w \text{ is a string}\}$$

Proof

On input $\langle B, w\rangle \in \Omega$, where

变快方法: keep track of all states NFA could be in at each point; $O(n)$ per input character

- $B = (\Sigma, Q, \delta, q_0, F)$ is a <u>NFA</u>
- $w \in \Sigma^*$ is input <u>string</u> to process on B.

1. Check if $\langle B, w\rangle$ is "proper" encoding. If not, reject.
2. Transform NFA B into an equivalent DFA C. 把NFA先变成DFA, $O(2^n)$ 耗时很长
3. Run TM for L_{DFA} on input $\langle C, w\rangle$ (需要把DFA转换成机器可读的编码 $O(m)$ $m=|w|$
4. If M accepts $\langle C, w\rangle$, accept; otherwise, reject. 做和上面 A_{DFA} 判断一样的事情

如果 A_{CFG} decidable，那 A_{PDA} 同样

L_{CFG} are decidable

Decision problem: Dose a CFG G generate a string w ?

$$L_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\} \subseteq \Omega$$

$$\Omega = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \text{ is a string}\}$$

$\langle G, w\rangle \in L_{CFG}$ if G generates w, $w \in L(G)$

$\langle G, w\rangle \notin L_{CFG}$ if G dosen't generate w, $w \notin L(G)$

CFGs are decidable

Recall

A context-free grammar $G = (V, \Sigma, R, S)$ is in **Chomsky normal form** if each rule is of the form

$$A \to BC \text{ or } A \to x \text{ or } S \to \varepsilon$$

- variable $A \in V$
- variables $B,C \in V - \{S\}$
- terminal $x \in \Sigma$.

Every CFG can be converted into Chomsky normal form

CFG G in Chomsky normal form is easier to analyze.

- for any string $w \in L(G)$ with $w \neq \varepsilon$ by derivation $S \overset{*}{\Rightarrow} w$ takes exactly $2|w| - 1$ steps. Base case $S \to w$ where w is singular letter takes 1 step. Additional one step to increase number of variables and another to realize it.
- $\varepsilon \in L(G)$ if G includes rule $S \to \varepsilon$.

CFGs are decidable

Proof

$S =$ "On input $\langle G, w \rangle$, where G is a CFG and w is a string: 先转换为乔姆斯基

1. Convert G to an equivalent grammar in Chomsky normal form.
2. List all derivations with $2n-1$ steps, where n is the length of w; except if $n = 0$, then instead list all derivations with one step.
3. If any of these derivations generate w, *accept*; if not, *reject*."

一共需要 2n-1步 派生

Or just run CYK algorithm to find all derivations

Emptiness of CFLs are decidable 上下文无关语言的空集问题

productive 指的是一个非终结符(non-terminal)能否通过一系列产生式(production rules)推导出至少一个终结符(terminal)，且不能只推导出 ε.

variable X is productive if
① $X \to w_1, \ldots, w_n$ all w_i are terminals
② $X = y_1 z_1 \ldots y_n z_n y_{n+1}$
y is terminal, z is productive variable

$E_{CFG} = \{\langle G \rangle \mid G$ is a CFG and $L(G)=\emptyset\}$

Proof idea:

Rule set is finite, and try exhaustively build parse tree from all potential list of terminals, and check whether we can reach the start symbol.

Proof

$R =$ "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G.
2. Repeat until no new variables get marked:
3. Mark any variable A where G has a rule $A \to U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, *accept*; otherwise, *reject*."

没有被 marked 代表 empty，所以 accept.

The Language L_{TM} is Turing-recognizable

$L_{TM} = \{\langle M, w\rangle : M$ is a Turing machine that accepts the string $w\}$

- If M accepts w, then $\langle M, w\rangle \in L_{TM}$
- If M doesn't accept w (reject or loop), then $\langle M, w\rangle \notin L_{TM}$

The language L_{TM} is Turing-recognizable.

Proof:

- A universal Turing machine U simulates M on w
  - If M accepts w, simulation will halt and accept
  - If M doesn't accept w (reject or loop), TM U either reject or loops.

The Language L_{TM} is undecidable

The language L_{TM} is undecidable.

$L_{TM} = \{\langle M, w\rangle : M$ is a Turing machine that accepts the string $w\}$

- The problem is that we don't really know whether the universal Turing machine will halt or not. Unlike all the machines we saw earlier, TM might run forever. 我们不确定 TM U 会不会停止，不像之前的机器，TM 可以一直循环
- Intuitively, it looks hard to find a decider for this problem.
- However, to show this is indeed undecidable is not trivial.

The Language L_{TM} is undecidable

The language L_{TM} is undecidable.

$L_{TM} = \{\langle M, w\rangle : M \text{ is a Turing machine that accepts the string } w\}$

there is a decider for A_{TM}
and derive some kind of contradiction

用反证法 对角化方法

Proof:
We will prove by contradiction and use the diagonalization method. We assume such a decider H exists, then show that the set of Turing machine is uncountable.

$$H(\langle M, w\rangle) = \begin{cases} accept & \text{if } M \text{ accepts } w \\ reject & \text{if } M \text{ does not accept } w. \end{cases}$$

The Language L_{TM} is undecidable

$L_{TM} = \{\langle M, w\rangle : M \text{ is a Turing machine that accepts the string } w\}$

Proof:
We will prove by contradiction and use the diagonalization method. We assume such a decider H exists, then derive a contradiction in terms of the countability of Turing machines.

$$H(\langle M, w\rangle) = \begin{cases} accept & \text{if } M \text{ accepts } w \\ reject & \text{if } M \text{ does not accept } w. \end{cases}$$

As the set of Turing machine is known to be countable, let's index them by M_i.
Now, it makes sense to ask the answer to $H(< M_i, < M_j >>)$ that is whether M_i accepts the description of M_j as input.

The Language L_{TM} is undecidable

$L_{TM} = \{\langle M, w\rangle : M$ is a Turing machine that accepts the string $w\}$

Proof continue:
As the set of Turing machine is known to be countable, let's index them by M_i.
Now, it makes sense to ask the answer to $H(< M_i, < M_j >>)$ that is whether M_i accepts the description of M_j as input.
We have this table

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$		$\vdots$			

Can we construct a Turing machine that is not in this list by the diagonalization method?
So that we will show the set of Turing machines is not countable, and we have a contradiction.

The Language L_{TM} is undecidable

$L_{TM} = \{\langle M, w\rangle : M$ is a Turing machine that accepts the string $w\}$

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$		$\vdots$			

Proof continue:
We know that this process will halt, as H is a decider.
Now, we construct a Turing machine D that flips the diagonal (**saying flipping is not enough, as the flipping needs to be done by a Turing machine**):

$D =$ "On input $\langle M\rangle$, where M is a TM:

D与H始值相反

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

Clearly, D is a decider and hence a Turing machine. It should be in the list.
Importantly, if H is not a decider, step 1 could loop forever, and D loop forever. Therefore, D does not really flip the diagonal.

The Language L_{TM} is undecidable

$L_{TM} = \{\langle M, w\rangle : M$ is a Turing machine that accepts the string $w\}$

$$
\begin{array}{c|ccccc}
 & \langle M_1\rangle & \langle M_2\rangle & \langle M_3\rangle & \langle M_4\rangle & \cdots \\
\hline
M_1 & accept & reject & accept & reject & \\
M_2 & accept & accept & accept & accept & \cdots \\
M_3 & reject & reject & reject & reject & \\
M_4 & accept & accept & reject & reject & \\
\vdots & & & \vdots & &
\end{array}
$$

Proof continue:

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

Clearly, D is a decider and hence a Turing machine. It should be in the list.

However, we know that by the diagonal flipping, it is not on the list as at least one value differs, and D is not in this list. It must be the case that Turing machines are not countable (so, we cannot do the numbering)

This is a contraction.

The Language L_{TM} is undecidable

$L_{TM} = \{\langle M, w\rangle : M$ is a Turing machine that accepts the string $w\}$

$$
\begin{array}{c|ccccc}
 & \langle M_1\rangle & \langle M_2\rangle & \langle M_3\rangle & \langle M_4\rangle & \cdots \\
\hline
M_1 & accept & reject & accept & reject & \\
M_2 & accept & accept & accept & accept & \cdots \\
M_3 & reject & reject & reject & reject & \\
M_4 & accept & accept & reject & reject & \\
\vdots & & & \vdots & &
\end{array}
$$

Alternative Proof:

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

Another common presentation of this proof is to ask directly:
Should D accepts <D>?
If D accepts <D>, in step 1 H accepts <D,<D>>, then in step 2 D rejects the <D>. 自我矛盾，因为
If D rejects <D>, in step 1 H rejects <D,<D>>, then in step 2 D accepts the <D>. D要与H相反.
We have a contradiction.

Instance of non Turing-recognizable languages.

Theorem: A language A is decidable if and only if it is Turing-recognizable and co-Turing-recognizable.

Proof: the only if part is simple, a decider always halts, and the decider accepts the language. For the complement of the language, a Turing machine accepts when the decider rejects and vice versa.

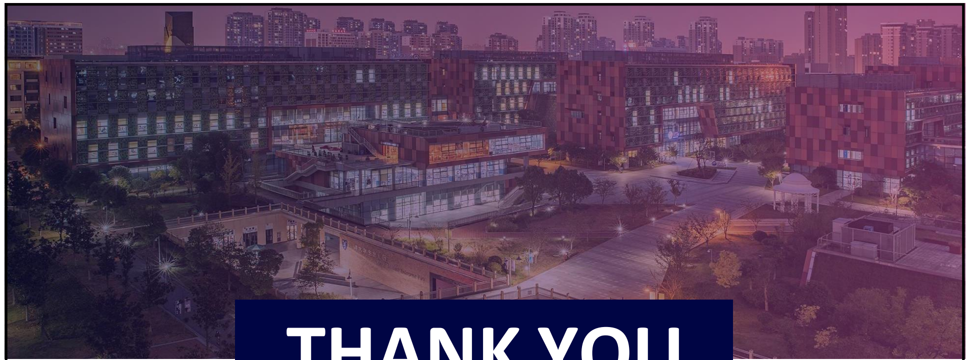
Now, for the if part, if both A and $\overline{A}$ are Turing-recognizable, we let M_1 be the recognizer for A and M_2 be the recognizer for $\overline{A}$. The following Turing machine M is a decider for A.

$M =$ "On input w:
1. Run both M_1 and M_2 on input w in parallel.
2. If M_1 accepts, *accept*; if M_2 accepts, *reject*."

Corollary $\overline{L_{TM}}$ is not Turing-recognizable.

Quick review

- Cantor's Diagonalization Method
- Church-Turing Thesis and Universal Turing Machine
- Examples of decidable languages
- Existence and instance of undecidable language and non-Turing recognizable language



THANK YOU

Xi'an Jiaotong-Liverpool University 西交利物浦大学

XJTLU | SCHOOL OF FILM AND TV ARTS

35