第 9 章 可判定性

1. 可判定语言
2. 不可判定性（Diagonalization & A_{TM}）
3. 可规约性（$HALT_{TM}$）
   - 调用可规约（Call Reducibility ）
   - 映射可规约（Mapping Reducibility）
   - 图灵可规约（Turing Reducibility）

Review

1. Computer Models (Automatas)

①FA $\Leftrightarrow$ RL(Regular Language)&RE

②PDA $\Leftrightarrow$ CFL(Context-free language)&CFG

③TM $\rightarrow$ TM-Recognizable ,TM-Decidable

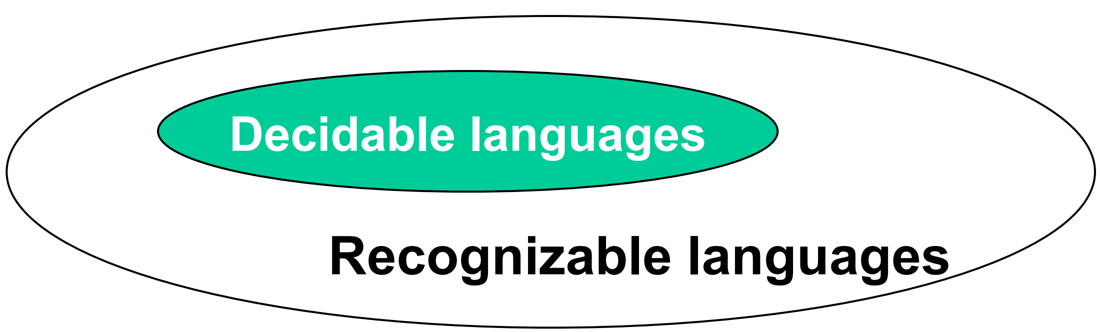
④Recursion Theorem

2. Church-Turing Thesis

Decide(判定) 与 Recognize(识别)有何区别?

Decidable : accept, reject (halting machine)

Recogniable: accept, reject, loop



Decidability

We are now ready to tackle the question 问题:

What can computers do and what not?
计算机 能做什么，不能作什么？ 不容易直接回答

转化为 考虑下列问题

Which languages are TM-decidable, Turing-recognizable, or neither?
哪些是图灵可识别，可判定或都不是? 容易多了

Assuming the Church-Turing thesis, these are fundamental properties of the languages.

Deciding Regular Languages

The acceptance problem for deterministic finite automata is defined by:

A_{DFA} = { <B,ω> | B is a DFA that accepts w }

注意，A_{DFA}是 DFA 和字符串的对子的集合，判定是指能对其一分为二，对子可编码成01串，所以，A_{DFA}是语言。

问题“**DFA B** 是否接受输入ω”与问题“**<B, ω >**是否是**A_{DFA}**的元素是相同的。

一些计算问题也可表示成检查语言的隶属问题，证明一个语言是否可判定的与证明一个计算问题是否可判定的是同一回事。

A_{DFA} is Decidable (Thm. 9.1)

Thm. 9.1 证明A_{DFA}是可判定的，即证明了问题“一个给定的有穷自动机是否接受一个给定的串”是可判定的。

Proof: Let the input <B,w> be a DFA with

$B=(Q, \Sigma, \delta, q_{start}, F)$ and $w \in \Sigma^*$.

The TM M1 performs the following steps:

1) Check if B and w are 'proper', if not: "reject"
2) Simulate B on w with the help of two pointers:
   $P_q \in Q$ for the internal state of the DFA, and
   $P_w \in \{0,1,\ldots,|w|\}$ for the position on the string.
   While we increase P_w from 0 to $|w|$, we change P_q according to the input letter w_{Pw} and the transition function value $\delta(P_q, w_{Pw})$.
3) If B accept w, then M accepts; otherwise M reject.

形式审查
内容审查

造TM，模拟DFA状态转移，放弃写功能和左移动功能，模拟DFA

Deciding NFA 定理9.2

Thm.9.2 The acceptance problem for nondeterministic FA A_{NFA} = { <B,w> | B is an NFA that accepts w } is a TM decidable language

注意， A_{NFA}是 NFA 和 语言的对子 的集合，TM 能对其一分为二，对子可编码成01串，所以 问题 是 语言.

Proof: Let the input <B,w> be an NFA with $B=(Q, \Sigma, \delta, q_{start}, F)$ and $w \in \Sigma^*$. 造TM M2如下：

```
bool  M2(A_NFA)
  {  把 A_NFA 转换成        //调用自动机确定化程序
          A_DFA ={ <C,w> | C is an DFA that accepts w }
     return ( M1(A_DFA);   // 调用上页结果的TM  M1
  }
```

Regular Expressions 定理9.3

Thm.9.3 The acceptance problem

A_{REX} = { <R,w> | R is a regular expression that can generate w }

is a Turing-decidable language.

语言与正则表达式对子的集合 是 识别与被识别 的关系

Proof Theorem 9.3. On input <R,w>:

1. Check if R is a proper regular expression and w a proper string //形式检查
2. Convert R into a DFA B // RE→DFA
3. Run earlier TM for A_{DFA} on <B,w>//调用上页结果

Emptiness Testing 空集合问题 Thm. 9.4

Thm.9.4 emptiness problem is decidable.

$E_{DFA} = \{\langle A\rangle \mid A \text{ is a DFA with } L(A) = \varnothing\}$ E-Empty

识别空语言的DFA（编码后）的集合，定出它的边界

在 E_{DFA} 之中的不识别任何语言，之外的识别一个语言。

意义：作为引理, 用于证明相等问题是可判定的。

Proof Idea:

$L(A) = \varnothing$, DFA A不接受字符串，也就是，从起始状态出发，到达不了可接受状态。

Proof for DFA-Emptiness

Algorithm for E_{DFA} on input $A=(Q,\Sigma,\delta,q_{start},F)$:

1) If A is not proper DFA: "reject" //形式审查
2) Make set S with initially $S=\{ q_{start} \}$
3) Repeat |Q| times:
   a) If S has an element in F then "reject"
   //传递到了接受态，传递路径被接受, 接受集非空
   b) Otherwise, add to S the elements that can be δ-reached from S via:
   "If $\exists q_i \in S$ and $\exists x \in \Sigma$ with $\delta(q_i,x)=q_j$, then q_j goes into S "
   //从S起，滚雪球或传销式地发展下家，发展进入S中

   If final $S \cap F = \varnothing$ "accept"
   //始终没发展接受态，不接受任何语言，则是空的

DFA-Equivalence Thm9.5

A problem that deals with two DFAs A and B:

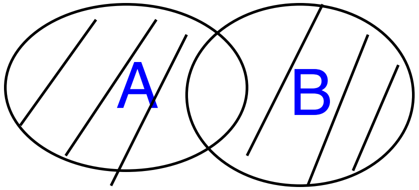
$EQ_{DFA} = \{<A,B> \mid L(A) = L(B) \}$ 异机识同语

EQ--Equal

Theorem 9.5: EQ_{DFA} is TM-decidable. 可判定

Proof: Look at the *symmetric difference* between the two languages:二者相等←→对称差为空

$$(L(A) \cap \overline{L}(B)) \cup (\overline{L}(A) \cap L(B))$$



对称差由RE的交、补、并合成，因而是RE.

问题转化为对称差的空问题判定（已经证明是可判定的）.

Proof Theorem 9.5 (cont.)

上页给了思想，这里还是给出算法（比TM说起来简单）

Algorithm on given <A,B>:

1) If A or B are not proper DFA: “reject”//形式审查
2) Construct a third DFA C that accepts the language (with standard transformations).

$$(L(A) \cap \overline{L}(B)) \cup (\overline{L}(A) \cap L(B))$$

3) Decide with the TM of the previous theorem whether or not $C \in E_{DFA}$
4) If $C \in E_{DFA}$ then “accept”; //对称差空，相等
   If $C \notin E_{DFA}$ then “reject” ”; //对称差不空，不等

Context-Free Languages

Similar languages for context-free grammars:

$A_{CFG} = \{ \langle G,w \rangle \mid G$ is a CFG that generates $w \}$
生成与被生成关系 问题　A--Accept

$E_{CFG} = \{ \langle G \rangle \mid G$ is a CFG with $L(G)=\varnothing \}$ 空问题
E--Empty

$EQ_{CFG} = \{ \langle G,H \rangle \mid G$ and H are CFGs with $L(G)=L(H) \}$ 相等问题

The problem with CFGs and PDAs is that they are inherently nondeterministic. 天生不确定

Chomsky NF

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

$A \rightarrow BC$(一分为二) or $A \rightarrow x$(终止符)

with variables $A \in V$ and $B,C \in V \setminus \{S\}$, and $x \in \Sigma$

For the start variable S we also allow "$S \rightarrow \varepsilon$"

简单而不失威力，理论推导时方便

Chomsky NF grammars are easier to analyze.

The derivation $S \Rightarrow^* w$ requires $2|w|-1$ steps (apart from $S \Rightarrow \varepsilon$). 重要：派生w的派生式长度固定。易检查。派生时步数虽多，但简单

Deciding CFGs (1)

Theorem 9.6: The language
A_{CFG} = { <G,w> | G is a CFG that generates w }
is TM-decidable. CFG生成关系 是可判定的

Proof: Perform the following algorithm:

1) Check if G and w are proper, if not "reject" //形式检查

//下面作内容检查:

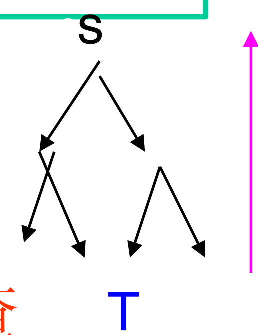
1) Rewrite G to G' in Chomsky normal form //简化
2) Take care of w=ε case via S→ε check for G'//先处理特例
3) List all G' derivations of length 2|w|–1//按长度检查派生式
4) Check if w occurs in this list; //是否有一个能派生出w
   if so "accept"; if not "reject" //定出受拒

Deciding CFGs (2)

Theorem 9.7: The language

E_{CFG} = { <G> | G is a CFG with L(G)=∅ }

is TM-decidable. CFG的空问题是可判定的



现在可用算法代替TM, 等价但比TM简洁

Proof: Perform the following algorithm:

1) Check if G is proper, if not "reject"//形式审查
2) Let G=(V,T,R,S), define set Σ = T //从叶子开始倒查
3) Repeat |V| times:
   - Check all rules $B \rightarrow X_1 \ldots X_k$ in R
   - If $B \notin T$ and $X_1 \ldots X_k \in \Sigma$ then add B to Σ //倒传销,找上家
4) If $S \in T$ then "reject", otherwise "accept" //根是上家，拒绝

Equality CFGs 意料之外的结果： 相等问题不可判定

What about the equality language

EQ_{CFG} = { <G,H> | G and H are CFGs
with L(G)=L(H) }? 相等问题

复习：DFA：空问题→对称差→相等问题 可判定

为什么这次不灵了？ 对称差用了 RL 对补、交 封闭。
而CFL 对补、交 不封闭，导致的不同。

Thm 9.8 each CFL is decidable

Thm. 9.8 每个上下文无关语言都是可判定的。

Proof // TM M2 调用TM S

设G是识别CFL A 的 CFG，由定理9.6，可以造TM S,

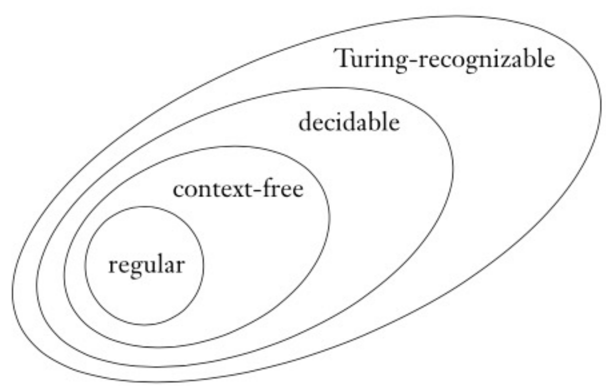
对w in A, S 可判定集合 { <G,w>|G是识别w 的 CFG},

即 S(<G,w>) 一定停机且返回 true 或 false.（不死循环）

造TM M2如下:

Bool M2(w)

{ return(S (<G,w>); }



Decidable

复　习：

1. 本章研究的主题是：算法求解问题的能力。结论是：有些问题是不可解的，即有些计算问题是不可判定的。

2. 计算问题可以用语言来描述

   ① 计算问题：检测一个特定的DFA B是否接受一个给定的串W。

   ② 语言A_{DFA}，包含了所有DFA及其接受的串的编码，其中

   A_{DFA} ={<B,W>|B is DFA, w is string , B accept w}。

   ③ 上述的计算问题可以用语言A_{DFA}来描述。

3. 证明一个计算问题是可判定的，与证明一个语言是可判定的是等价的。

Halting Problem

下列问题可判定

A_{DFA} = { <B,w> | B is a DFA that accepts w }

A_{CFG} = { <G,w> | G is a CFG that generates w }

are TM decidable.

问题是：

1. A_{TM} = {<M,w> | M is a TM that accepts w } is TM-Decidable or TM-Recognizable?
2. Is one TM **U** capable of simulating all other TMs?(Universal TM)

A_{TM} 又称为接受问题或停机问题，接受问题可被识别，但不能被判定。

Universal TM

引入通用图灵机的直观概念

Win中模拟DOS上的dir WinExec(“command.com/C”,”dir”);

Win 是TM, Dos 是TM, Dos可以编码成为串“M”

仿真时，Win相当于通用图灵机

Win(“M”,”dir”)

{分配M所需的空间S，

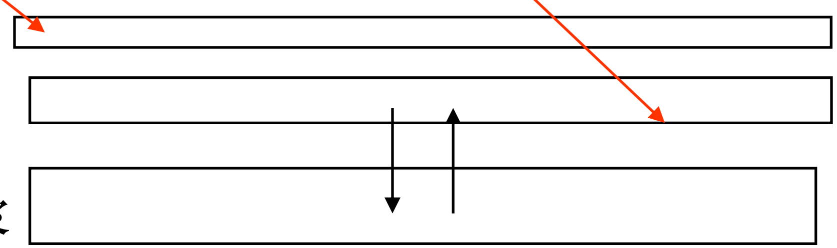
把“M”复制到S上去；

在Win的监控下，在S上运行DOS,运行 dir，

善后，退出；

}   用3带机

1. Win 仿真控制带
2. 被模拟机带S: Dos
3. 演算带,Buff当前内容



Universal TM

Given a description <M,w> of a TM M and input w, can U simulates M on w?

构造一个通用图灵机

We can do so via a universal TM U (2-tape):

1) Check if M is a proper TM
   Let $M = (Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$
2) Write down the starting configuration
   $< q_0w >$ on the second tape
3) Repeat until halting configuration is reached:
   - Replace configuration on tape 2 by next configuration according to δ
4) "Accept" if q_{accept} is reached; "reject" if q_{reject}

简言之：bool U(M,w)

{ return(M (w)) ;} //如果M不死循环，U也不死循环

A_{TM} is decidable?

$A_{TM} = \{\langle M,w\rangle \mid M \text{ is a TM that accepts } w\}$

is TM-recognizable, but can we also *decide* it ?

The problem lies with the cases when M does not halt on w. In short: <u>the halting problem</u>.

问题焦点 ： M 死循环的判断。所以A_{TM}又称*停机问题*

精确的停机问题应该是:

$$HALT_{TM}=\{\langle M,w\rangle \mid \text{TM } M \textbf{ halts} \text{ for } w\}$$

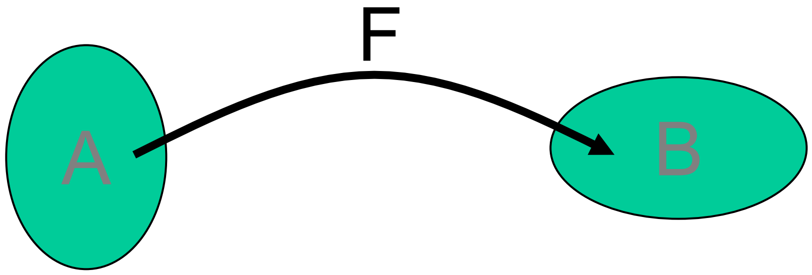
We will see that this is an insurmountable problem: in general , one cannot decide if a TM will halt on w or not, hence A_{TM} is undecidable.

先揭谜底：停机问题不可判定， 从而A_{TM}不可判定

为证明它，先补充一系列预备知识，

Mappings and Functions 用映射比较集合大小

The function $F:A \rightarrow B$ maps one set A to another set B:



F is one-to-one (injective 内射，不同源有不同像，源<一像) if every $x \in A$ has a unique image $F(x)$: If $F(x)=F(y)$ then $x=y$.

F is onto (surjective满射) if every $z \in B$ is 'hit' by F:If $z \in B$ then there is an $x \in A$ such that $F(x)=z$.

F is a correspondence (bijection双射) between A and B if it is both one-to-one and onto. 规模相同

Cardinality

A set S has k elements if and only if there is a bijection possible between S and {1,2,…,k}.

S and {1,…,k} have the same cardinality（集的势）.

If there is a surjection possible from {1,…,n} to S, then $n \geq |S|$.

We can generalize this way of comparing the sizes of sets to infinite ones.

Countable Infinite Sets

A set S is infinite if there exists a surjective(满射) function F:S→N. 基数>=自然数集数

"The set N has not more elements than S."

A set S is countable if there exists a surjective function F:N→S "The set S has not more elements than N."

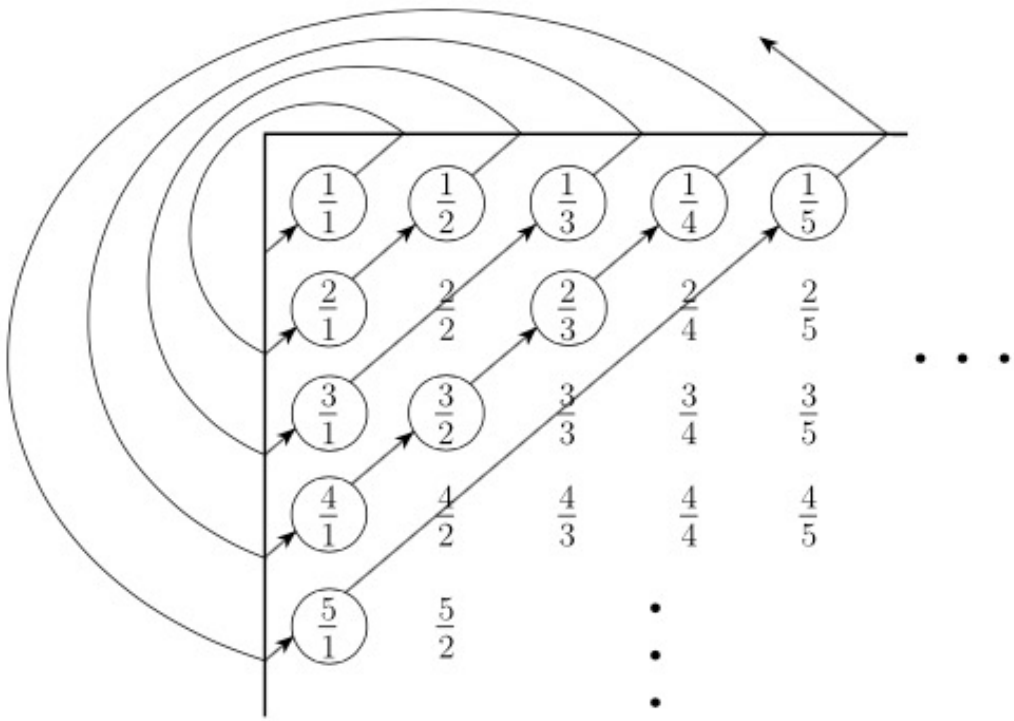
有限集可数，与自然数集合等势可数

A set S is countable infinite if there exists a bijective function F:N→S. 可数无穷，与N 等势

"The sets N and S are of equal size."

Countable Infinite Sets

有理数集合可数 每个 n/m 都能被数到



Diagonalization 对角线方法

Theorem 9.9 R is uncountable

n	$f(n)$
1	$3.\underline{1}4159\ldots$
2	$55.5\underline{5}555\ldots$
3	$0.12\underline{3}45\ldots$
4	$0.500\underline{0}0\ldots$
$\vdots$	$\vdots$

$$x = 0.4641\ldots$$

x is not f(n) for any n because it differs from f(n) in the nth fractional digit.

Counting TMs 有多少图灵机

Corollary 9.18 Some languages are not Turing-recognizable.

Observation: Every TM has a finite description; there is only a countable number of different TMs. (A description <M> can consist of a finite string of bits, and the set $\{0,1\}^*$ is countable.)

C语言程序，只有可数个，文章只有可数篇，同理，
图灵机由有限个字符描述，编码后按字典序排，只有可数个。

Our definition of Turing recognizable languages is a mapping between the set of TMs $\{M_1, M_2, \ldots\}$ and the set of languages $\{L(M_1), L(M_2), \ldots\} \subseteq \mathcal{P}(\Sigma^*)$.

Counting Languages

There are uncountable many different languages over the alphabet Σ={0,1} (the languages L ⊆ {0,1}*). With the lexicographical ordering ε,0,1,00,01,… of Σ*, every L coincides with an infinite binary sequence via its characteristic sequence（特征序列） χ_L.

Example for L={0,00,01,000,001,…} with χ_L= 0101100…

语言　特征串（有X，无 空格），　字符串（字典序）

Σ^*	ε	0	1	00	01	10	11	000	001	010	$\cdots$
L		X		X	X			X	X	X	$\cdots$
χ_L	0	1	0	1	1	0	0	1	1	1	$\cdots$

Counting TMs and Languages

There is a bijection between the set of languages over the alphabet $\Sigma=\{0,1\}$ and the uncountable set of infinite bit strings $\{0,1\}^N$. There are uncountable many different

languages $L \subseteq \{0,1\}^*$. 语言 不可数

- Hence there is no surjection（满射） possible from the countable set of TMs to the set of languages.

Specifically, the mapping L(M) is not surjective.

但图灵机（程序、系统）只有可数个

Conclusion: There are languages that are not Turing-recognizable. (A lot of them.)

不可识别的的语言 不但存在， 而且占了绝大部分。

停机问题A_{TM} 不可判定 （A - Accept, 应称为接受问题）

停机问题： Consider again the acceptance language
A_{TM} = { <M,w> | **M is a TM that accepts w** }.
这里，集合：一切合乎条件的元素, 包括A_{TM}自己，自己判定自己， 突破点就在就在这里，

Proof that A_{TM} is not TM-decidable (Thm. 9.19)
(反证法) Assume that TM H decides A_{TM}:

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

用C语言描述: bool H(M,w) { return(M(w)； } //组件调用

From H we construct a new TM D that will get us into trouble… 拟造D,导出矛盾

Proving Undecidability

窍门：把M自己搅进去，让他自己判定自己，导出矛盾

The TM D works as follows on input <M> (a TM):

1) Run H on <M,<M>> //让M的编码串作自己的输入

2) Disagree with the answer of H //相当于对角线反码

(The TM D always halts because H always halts.)

In short: $D\langle M\rangle = \begin{cases} \text{"accept" if } H \text{ rejects } \langle \mathrm{M}, \langle \mathrm{M}\rangle\rangle \\ \text{"reject" if } H \text{ accepts } \langle \mathrm{M}, \langle \mathrm{M}\rangle\rangle \end{cases}$

Hence: $\mathrm{D}\langle \mathrm{M}\rangle = \begin{cases} \text{"accept" if M does not accept } \langle \mathrm{M}\rangle \\ \text{"reject" if M does accept } \langle \mathrm{M}\rangle \end{cases}$

D也是一切中的一个，Now run D on <D> (“on itself”)…

Proving Undecidability

Result : 矛盾

$$D\langle D\rangle = \begin{cases} \text{"accept" if D does not accept } \langle D\rangle \\ \text{"reject" if D does accept } \langle D\rangle \end{cases}$$

This does not make sense: D only accepts if it rejects, and vice versa.
(Note again that D always halts.)

Contradiction: **A_{TM} is not TM-decidable**.

This proof used diagonalization implicitly…

Review of Proof (1)

‘Acceptance behavior’ of M_i on $<M_j>$

图灵机 输入串

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	accept		accept		
M_2	accept	accept	accept	accept	
M_3					$\cdots$
M_4	accept	accept			
$\vdots$			$\vdots$		$\ddots$

Review of Proof (2)

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

'Deciding behavior' of H on $<M_i,<M_j>>$，拟用对角线上反码构造图灵机D

Review of Proof (3)

相反 相反 相反 相反

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

D也是一切合乎条件的TM 中的一个，所以也在其中

Review of Proof (3)

相反 相反 相反 相反

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*			
M_2	*accept*	*accept*	*accept*	*accept*			
M_3	*reject*	*reject*	*reject*	*reject*	$\cdots$		
M_4	*accept*	*accept*	*reject*	*reject*			
$\vdots$			$\vdots$		$\ddots$		
D	*reject*	*reject*	*accept*	*accept*	$\cdots$	$\boxed{?}$	
$\vdots$							

Contradiction for D on input <D>. 接受、拒绝都矛盾

TM-Unrecognizable

A_{TM} is not TM-decidable, but it is TM-recognizable.
What about a language that is not recognizable?

Theorem 9.20: If a language A
A 可判定 ⟺ A and Ā is TM recognizable

Proof: Run the recognizing TMs for A and Ā in parallel on input x. Wait for one of the TMs to accept. If the TM for A accepted: “accept x”; if the TM for Ā accepted: “reject x”.
并行或分时并发识别A和Ā，其中之一结束就结束

TM-Unrecognizable

Theorem 9.20: If a language A

A 可判定 ⟺ A and Ā is recognizable

Proof: ⟹ 显然。

⟸ 并行或分时并发 识别A和Ā，有一个结束就结束

给定TM M1 定义 步进图灵机

```
Bool  Step_M1(w,n)
{  在M1运行n步的基础上（状态，带位置）再运行一步
    if M1到达终止状态  return(true); else return false;
}
```

设M2是识别补集的TM 类似地定义 Step_M2(w,n)

下面是 判定A 的并行TM M :

```
bool M(w)
 { n=0;  stop=false; while (! stop)
       { stop=Step_M1(w,n) || !Step_M2(w,n)); n++;}
 }
```

A接受W, 则 Step_M1(w,n）为真

TM-Unrecognizable

Theorem 9.20: If a language A

A 可判定 ⟺ A and Ā is recognizable

Proof: ⟹ 显然。

⟸ 并行或分时并发 识别A和Ā，有一个结束就结束

给定TM M1 定义 步进图灵机

```
Bool  Step_M1(w,n)
{ 在M1运行n步的基础上（状态，带位置）再运行一步
   if M1到达终止状态  return(true); else return false;
}
```

设M2是识别补集的TM 类似地定义 Step_M2(w,n)

下面是 判定A 的并行TM M :

```
bool M(w)
 { n=0;  stop=false; while (! stop)
      { stop=Step_M1(w,n) || !Step_M2(w,n)); n++;}
    return stop;
 }
```

A拒绝W, 则

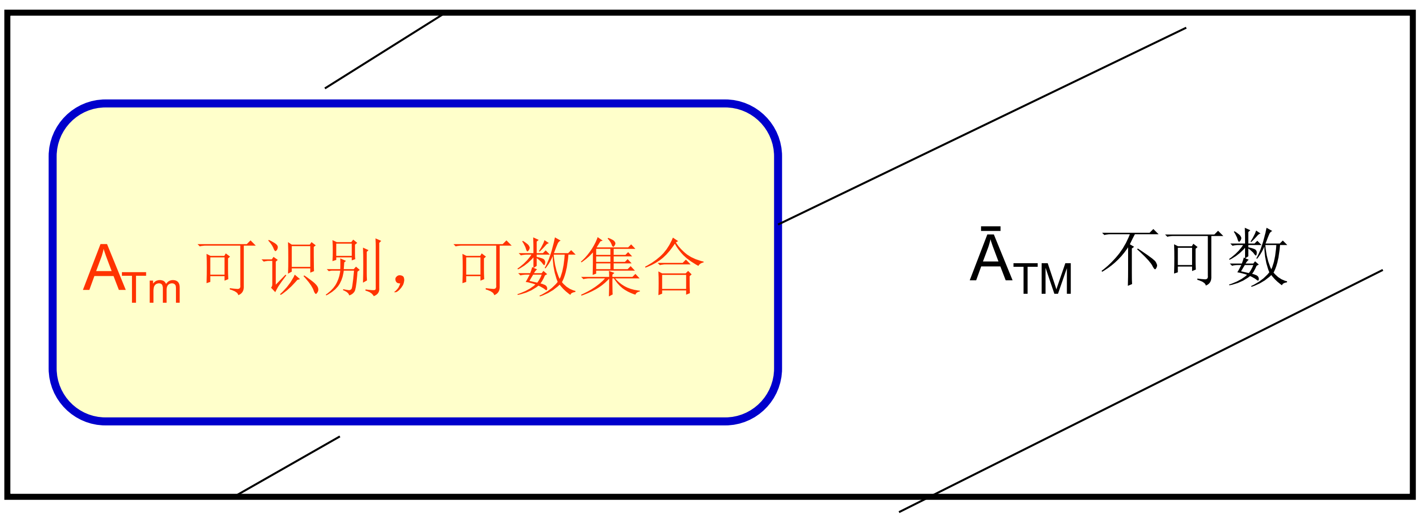
！ Step_M1(w,n）为真

$\bar{A}_{TM}$ is not TM-Recognizable
停机问题的补问题是不可识别的

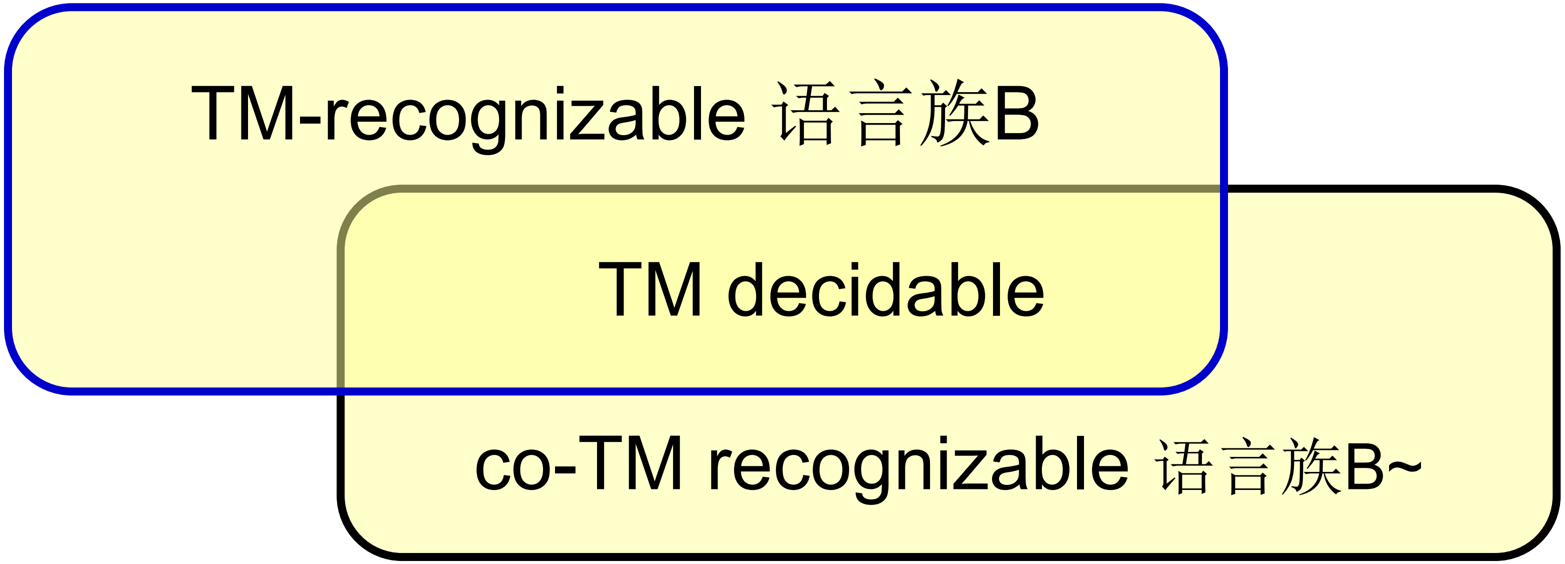
反证法：已知 A_{TM} 可识别，如果其补集可识别，则由上面定理。推出停机问题可判定，与前面结果矛盾。

直观：语言总集不可数，可识别的集合A 是可数集合，其补集是不可数的，集合太大，当然不可识。

We call languages like $\bar{A}_{TM}$ <u>co-TM recognizable</u> 它不一定是可识别的



补问题



Things that TMs Cannot Do:

The following languages are also unrecognizable:

$E_{TM} = \{ \langle G \rangle \mid G \text{ is a TM with } L(G)=\varnothing \}$ 空问题

$EQ_{TM} = \{ \langle G,H \rangle \mid G \text{ and } H \text{ are TMs with } L(G)=L(H) \}$ 相等问题

To be precise:

- E_{TM} is co-TM recognizable
- EQ_{TM} is not even co-Turing recognizable

小结与回顾

1. Deciding RL properties
2. Deciding context-free languages
3. The Halting Problem
4. Countable and uncountable infinities
5. Diagonalization arguments

Reducibility 可归约性

归约的直观解释：调用问题

A官 , B兵, 对应C程序(TM)： Prog_官 (w) ,Prog_兵(w)

如果官调兵 如下：

```
Prog_官(w)
{ ……
   …..                    //简单计算
   Prog_兵(w)
   ……                     //简单计算
}
```

则称官 规约为 兵， 从计算能力比较有 官 >= 兵

主调 被调

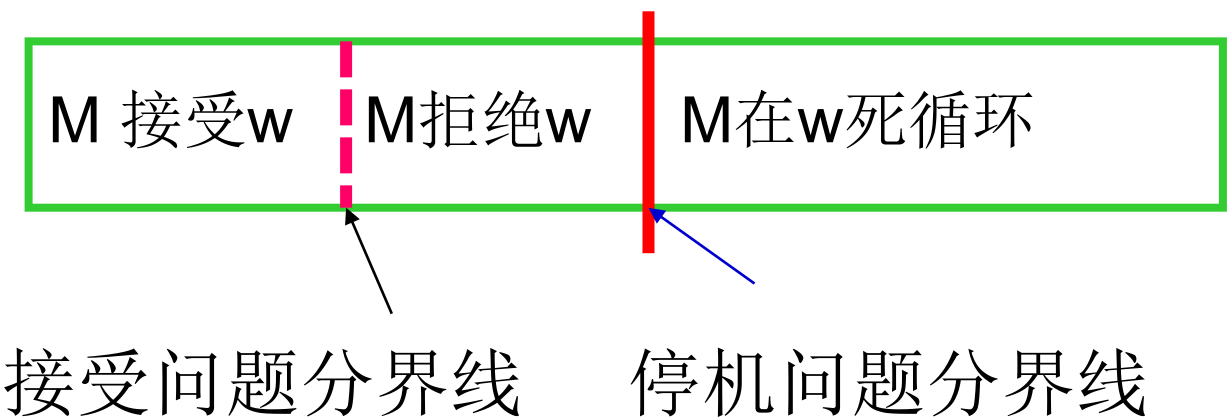
- 如果 B兵能识别语言 L则A官也能
- 逆否定理 如果A官不能识别L, 则B兵也不能

Halting Problem Revisited

原来把 A_{TM} = { <M,w> | the TM M 识别 w }
称为 停机问题 是因为它与下列名副其实的停机问题很近
A_{TM} 应该称为接受问题, A—Accept

名副其实的停机问题:

Theorem 9.21: The ‘halting problem’ language
$HALT_{TM}$ = { <M,w> | the TM M halts on input w }
is undecidable (but of course recognizable).



同济大学 TONGJI UNIVERSITY

Halting Problem Revisited

Theorem 9.21: The ‘halting problem’ language $HALT_{TM}$ = { <M,w> | the TM M halts on input w } is undecidable (but of course recognizable).

Proof: 反证法

反设存在判定器 Deter_Halt 判定 $HALT_{TM}$，则可证明：

TM Deter_Accept decides A_{TM}。

```
bool Deter_Accept(< M,w> ) //接受问题
  { if (Deter_Halt(<M, w>) )   //停机问题
       return ( M (w))   //M接受W,返回true, 否则 false
  }
```

以前已证明接受问题不可判定（对某些w,会死循环),如停机问题能判定，则与以前结论矛盾，证毕。

Halting Problem Revisited

Theorem 9.21: The ‘halting problem’ language $HALT_{TM}$ = { <M,w> | the TM M halts on input w } is undecidable (but of course recognizable).

Proof: Let R be a TM that decides $HALT_{TM}$.

The following TM S decides A_{TM}:

On input <M,w> ***run*** R to decide halting

1. If R rejected <M,w> , then “reject” .
2. If R accepted <M,w>

   then copy (reject/accept) output of M on w.

(Note that this TM S always produces an output.)

A_{TM} is undecidable, hence such a R cannot exist.

问题：规约体现在哪里？

Deciding Equality

Theorem 9.22: The language of non-accepting TMs
$E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM with } L(M)=\varnothing \}$
is not decidable (but co-TM recognizable).
空接受问题不可判定(上课板书)

Theorem 9.23: $EQ_{TM} = \{ \langle M1,M2 \rangle \mid M1,M2 \text{ TMs}, L(M1)=L(M2) \}$ is undecidable
相等问题不可判定(上课板书)

Deciding Equality

Almost any language property of Turing machines is undecidable:

Theorem 9.24

$Regular_{TM}$ = { <M> | L(M) is a regular language }
一切识别正则语言的图灵机集合

$Finite_{TM}$ = { <M> | L(M) is a finite language }
一切识别有限语言的图灵机集合

CFG_{TM} = { <M> | L(M) is a CFG language }
一切识别CFG语言的图灵机集合

Are undicidable

Review

1、归约的目的在于：

将一个问题转化为另一个问题；且用第二个问题的解来解决第一个问题。比如：在城市中认路可归约为得到一张地图的问题。

2、归约的应用（A问题可归约到B问题）：

- 如果B是可判定的，则A也是可判定的；
- **如果A是不可判定的，则B也是不可判定的；（主要的应用）**

Reveiw

3、A_{TM}是不可判定的，但， A_{TM}是可识别的：

U=“对于输入<M,w>，其中M是TM，w是字符串：

① 在输入w上模拟M；

② 如果M接受，则接受；如果M拒绝，则拒绝。”

一旦M在w上死循环，U无法预知，且只能陷入死循环，所以A_{TM}是可以识别的，但不可判定（采用对角化方法可证明）。

4、$HALT_{TM}$是不可判定的：

设$HALT_{TM}$是可判定的，则可证明A_{TM}是可判定的，出现矛盾。所以$HALT_{TM}$是不可判定的。

5、E_{TM}，EQ_{TM}，$REGULAR_{TM}$都是不可判定的。


Thank You !