Introduction to the Theory of Computation

Outline for today

1、FA and PDA

2、Turing Machine

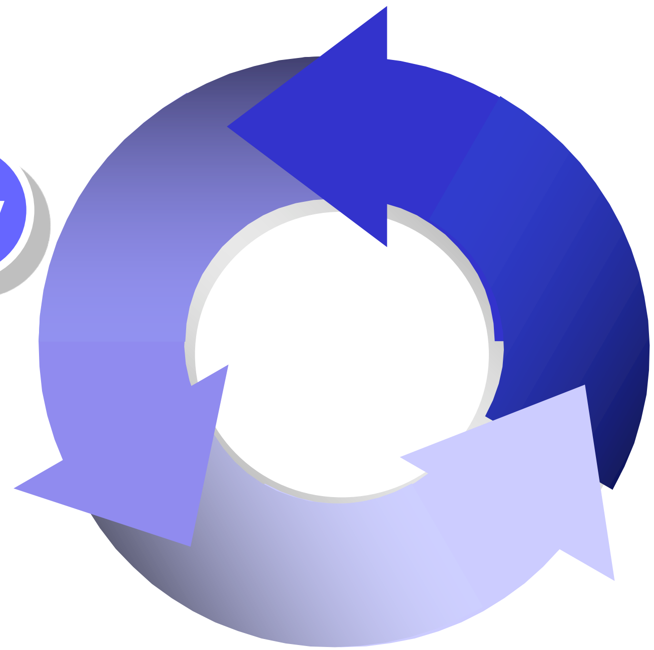
3、Variants of Turing Machine

4、The Definition of Algorithm

5、How to describe the Algorithms

Content of courses

Automata Theory

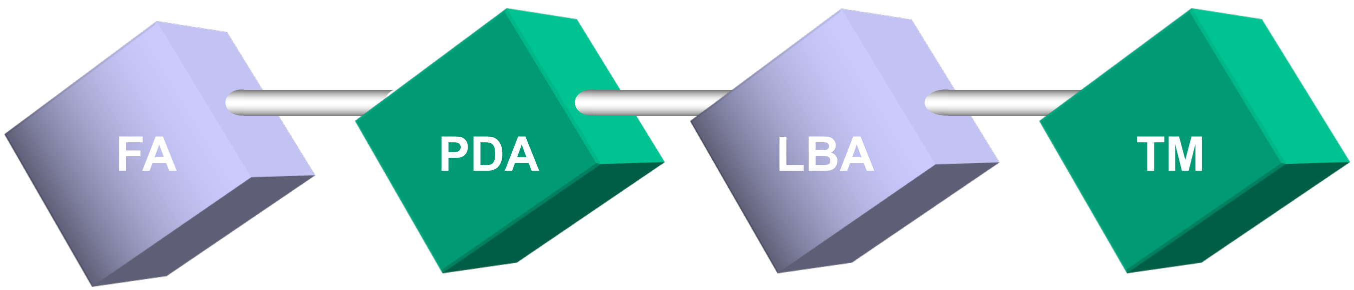


Computability Theory

Complexity Theory

Content of courses

1. Automata Theory



正则语言（RL），3型	前后文无关语言（CFL），2型	前后文有关语言（CSL），1型	递归可枚举（r.e.），0型

Content of courses

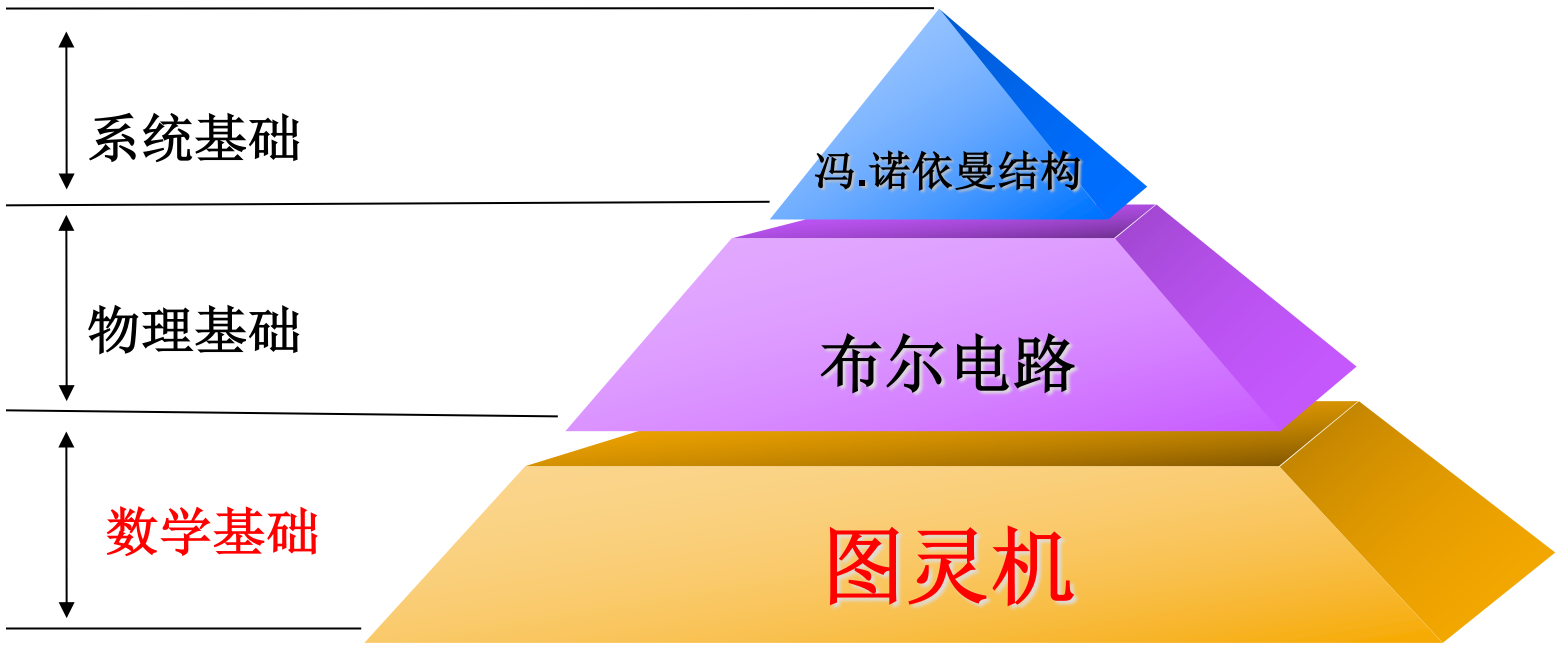
2. Computability Theory

In computability theory, the classification of problems is by those that are solvable and those that are not.

3. Complexity Theory

In complexity theory, the objective is to classify problems as easy and hard ones, whereas

■ 图灵机概念的引入



电子计算机的三大基础

今天所有的计算机，都是图灵机的实例，都建立在冯.诺依曼结构之上，都由若干电子器件组合而成的。

Turing Machine

- On computable Number, 1936
  - 这篇奠基之作其实是回答德国大数学家David Hilbert在世界数学家大会上提出的“23个数学难题”中的一个问题：“是否所有的数学问题在原则上都是可解的”
  - 图灵认为“有些数学问题是不可解的”
  - 图灵机只是在这篇论文的一个脚注中顺便提出的

Endnotes

8. It is most natural to construct first a choice machine (§2) to do this. But it then easy to construct the required automatic machine. We can suppose that the choices are always choices between two possibilities 0 and 1. Each proof will then be determined by a sequence of choices *i*1, *i*2, …, *i*n (*i*1 = 0 or 1, *i*2 = 0 or 1, …, *i*n = 0 or 1), and hence the number 2*n* + *i*1 25+1 + *i*2 25-2+...+ *i*n, completely determines the proof. The automatic machine carries out successively proof 1, proof 2, proof 3, ….

- **Computer Models**
  - Finite Automata: a small amount of memory
  - Pushdown Automata: an unlimited Stack
  - Turing Machine: unlimited and unrestricted memory
- **Turing & Church**

In 1936, “*On Computable Numbers, with an application to the Entscheidungs problem*” . at the same time, Alonzo Church published similar ideas and results. However, the Turing model has become the standard model in theoretical computer science.

Alonzo Church (1903–1995)

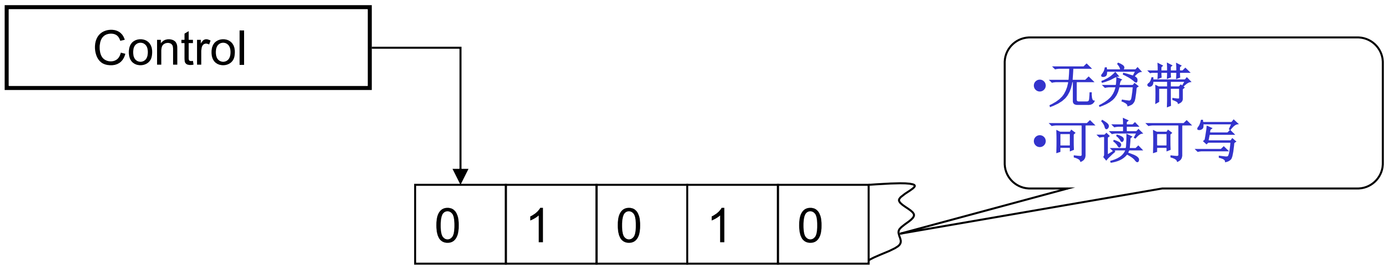


Figure 8.1 Schematic of a Turing machine

Differences between FA and TM:

1. The tap is infinite. 内存无限
2. TM can both read from the tap and write on the tap.
3. The control head can move both to the left and to the right.
4. TM ***immediately*** halt, once to accepting state or rejecting state.

(Computation's results: accept , reject or loop.)

Definition 8.1

A Turing machine M is defined by a 7-tuple $(Q,\Sigma,\Gamma,\delta,q0,q_{accept},q_{reject})$, with

- Q: finite set of states 状态集合
- Σ: finite input alphabet (without “⊔”) 输入字符
- Γ: finite tape alphabet with $\{⊔\} \cup \Sigma$ 带字符
- q_0: start state ∈Q 开始状态
- q_{accept} : accept state ∈Q 接受状态
- q_{reject} : reject state ∈Q 拒绝状态
- δ: the transition function 转移函数

either left or right

$$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

当前状态，当前字符，转移后状态，写，移

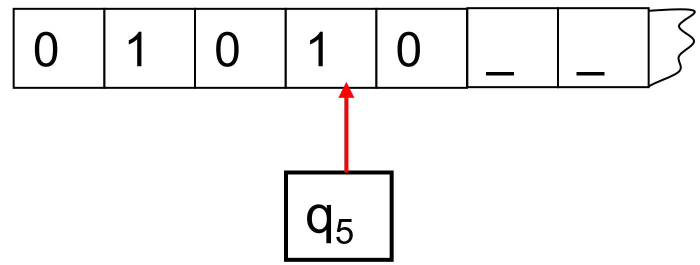
δ(q_i, b) = (q_j, c, L/R) , $q_i \neq q_{accept}$，$q_i \neq q_{reject}$

Configuration(格局)

1.The current state q ∈Q

2.The current tap contents Γ^*

3.The current head location



Configuration as : $010\mathbf{q_5}10$

Tree kinds of Configurations:

1. starting configuration on input w: “q_0w” 初始格局
2. accepting configuration: “$uq_{accept}v$” 接受格局
3. rejecting configuration: “$uq_{reject}v$” 拒绝格局

The accepting and rejecting configurations are the halting configurations（停机格局）.

计 算 的 定 义

Yields （产生）

Let u,v∈Γ* ; a,b,c∈Γ; q_i,q_j∈Q, and M is a TM with transition function δ. We say that the configuration "uaq$_i$bv" yields the configuration "uacq$_j$v", if and only if: δ(q_i,b) = (q_j,c,R).

如果格局序列:C_1,C_2,…使得C_1 yields C_2,C_2 yields C_3,…C_{n-1} yields C_n,而且当序列有穷时,而且最后一个格局C_n是停机格局,则称这个序列（或格局演化的过程）是Turing机的一个计 算（或计算过程）。

关于计算的定义（第一次）秘❤️

格局的作用：提供了一种分析问题的方法，如考察一个问题的可计算性等。

The collection of strings that M accepts is the language of M, denoted L(M).

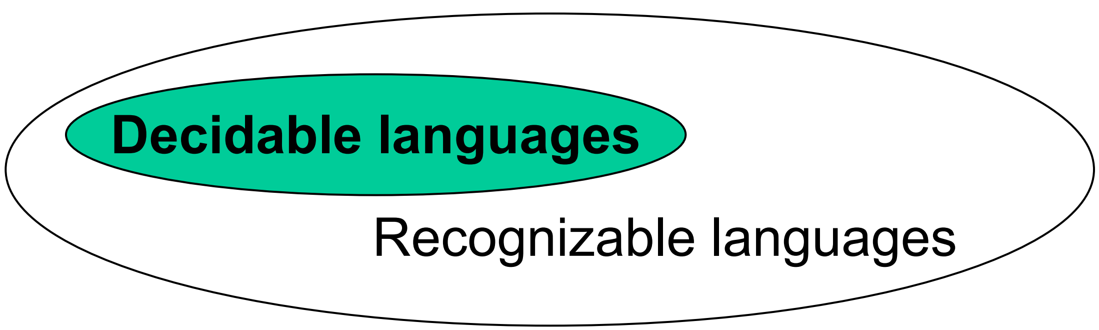
Def. 8.2 A language L is Turing-recognizable (图灵可识别) if and only if there is a TM recognize it.

1. Also called: a recursively enumerable language(递归可枚举语言).
2. Note: On an input $\omega \notin L$, the machine M can halt in a rejecting state, or it can ‘loop’ indefinitely.
3. 图灵可识别的结果是: accept, reject or **loop**.

Turing Machine

Def. 8.3 If a language can be decided by some Turing Machine , then call the language Turing-decidable or decidable.（图灵可判定 或 可判定）

1. Also called: a recursive language(递归语言) .
2. Decider is a TM that halt on all inputs, never loops.
3. Decide(判定) 与 Recognize(识别)有何区别?



EXP8.1：Design a TM M_1, decides L＝｛0^{2^n}|n≥0｝

//用C语言模拟TM, 注意不要超标使用资源

```
bool M( j)  //j代表字符串的长度
{
  if (j==0) return false ;
  if ( j==1) return true;
  if ( j mod 2==0) return M(j/2) //这里有点超前，使用了递归
  else if (j>1)
          return false ;    //注意无论 j 为何值，总有结果
}
```

High level description of TM

M_1 =“ On input string ω:

1.Sweep left to right across the tape, crossing of every other 0;   (删除一半的0)

2.If in stage 1 the tape contained a single 0, **accept** ;

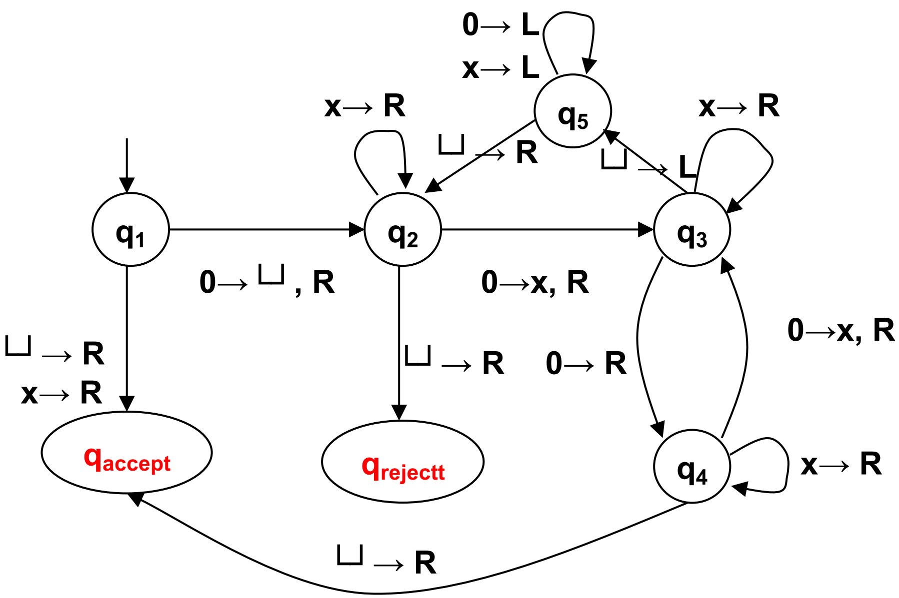
3.If in stage 1 the tape contained more than a single 0 and the number of 0s is odd, **reject**;

4.Return the head to the left-hand end of the tape.

5.Go to stage 1. ”

Formal Definition of M_1:

$Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$, $\Sigma = \{0\}$, $\Gamma = \{0, x, \sqcup\}$



输入0000后的格局序列：

$q_1 0000$	$\sqcup q_5 \mathrm{x0x}\sqcup$	$\sqcup \mathrm{x} q_5 \mathrm{xx}\sqcup$
$\sqcup q_2 000$	$q_5 \sqcup \mathrm{x0x}\sqcup$	$\sqcup q_5 \mathrm{xxx}\sqcup$
$\sqcup \mathrm{x} q_3 00$	$\sqcup q_2 \mathrm{x0x}\sqcup$	$q_5 \sqcup \mathrm{xxx}\sqcup$
$\sqcup \mathrm{x0} q_4 0$	$\sqcup \mathrm{x} q_2 \mathrm{0x}\sqcup$	$\sqcup q_2 \mathrm{xxx}\sqcup$
$\sqcup \mathrm{x0x} q_3 \sqcup$	$\sqcup \mathrm{xx} q_3 \mathrm{x}\sqcup$	$\sqcup \mathrm{x} q_2 \mathrm{xx}\sqcup$
$\sqcup \mathrm{x0} q_5 \mathrm{x}\sqcup$	$\sqcup \mathrm{xxx} q_3 \sqcup$	$\sqcup \mathrm{xx} q_2 \mathrm{x}\sqcup$
$\sqcup \mathrm{x} q_5 \mathrm{0x}\sqcup$	$\sqcup \mathrm{xx} q_5 \mathrm{x}\sqcup$	$\sqcup \mathrm{xxx} q_2 \sqcup$
		$\sqcup \mathrm{xxx}\sqcup q_{accept}$

问题：

1. 如何确定输入带的最左端？
2. 上面的例子是一个计算吗？

Note:

0→x, R **means read 0, write x, move Right**

EXP2: TM M_2 decides language $C=\{a^i b^j c^k \mid i \times j=k,\ \ i,j,k \geq 1\}$

M_2 = "On input string ω:

1. Scan the input from left to right to determine whether it is a member of $a^+b^+c^+$ and reject if it isn't.
2. Return the head to the left-hand end of the tape. (可以采用标记法)
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and the c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, reject.
4. Restore the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's have been crossed off, determine whether all c's also have been crossed off. If yes, accept; otherwise, reject.

A k-tape Turing machine(多带图灵机) M has k different tapes and read/write heads. It is thus defined by the 7-tuple $(Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$, with

- Q finite set of states
- Σ finite input alphabet (without "$\sqcup$")
- Γ finite tape alphabet with $\{\sqcup\} \cup \Sigma \subseteq \Gamma$
- q_0 start state $\in Q$
- q_{accept} accept state $\in Q$
- q_{reject} reject state $\in Q$
- δ the transition function

$$\delta: Q\backslash\{q_{accept},q_{reject}\} \times \Gamma^k \to Q \times \Gamma^k \times \{L,R,S\}^k$$

Theorem 3.13: For every multi-tape TM M, there is a single-tape TM M' such that L(M)=L(M').
Or, for every multi-tape TM M, there is an equivalent single-tape TM M'.

多带机与单带机等价
增加存储和数组(多带)只提速和简化， 无本质改变

Proving and understanding these kinds of results, is essential for appreciating the Turing machine model. 称为稳健性

From this theorem Corollary c3.15 follows: A language L is TM-recognizable if and only if some multi-tape TM recognizes L. 以后可用 多带机 作题， 简单多了

Theorem 3.13: For every multi-tape TM M, there is a single-tape TM M' such that L(M)=L(M'). Or, for every multi-tape TM M, there is an equivalent single-tape TM M'.

Proving and understanding these kinds of ***robustness*** *results, is essential for appreciating the power of the Turing machine model.*

稳健性
多带机与单带机等价
增加存储和数组(多带)只提速和简化， 无本质改变

From this theorem Corollar … L
is TM-recognizable if and o…
recognizes L. 以后可用 多…

同濟大學
TONGJI UNIVERSITY

Theorem 3.13: For every multi-tape TM M, there is a single-tape TM M' such that L(M)=L(M').
Or, for every multi-tape TM M, there is an equivalent single-tape TM M'.

意义：以后可用多带机作题，简单多了

Proving and understanding these kinds of robustness results, is essential for appreciating the power of the Turing machine model.

From this theorem Corollary 3.15 follows: A language L is TM-recognizable if and only if some multi-tape TM recognizes L.

思路： 两个模型等价 ⇔ 两个模型可以相互模拟

1. 模拟结构

造单带机模拟多带机 （多带机模拟单带机不需证明）

Let $M=(Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$ be a k-tape TM.

Construct 1-tape M' with expanded $\Gamma' = \Gamma \cup \underline{\Gamma} \cup \{\#\}$

第1道 第k道格局

Represent M-configuration

$$u_1 q_j a_1 v_1, \quad u_2 q_j a_2 v_2, \quad \ldots, \quad u_k q_j a_k v_k$$

by M' configuration,

$$q_j \# u_1 \underline{a}_1 v_1 \# u_2 \underline{a}_2 v_2 \# \ldots \# u_k \underline{a}_k v_k$$

分带符 #, K道上当前字符

2. 模拟动作

1.On input $w=w_1\ldots w_n$, the TM M' does the following: prepare initial string: $\#\underline{w}_1\ldots w_n\#\underline{\ }\#\cdots\#\underline{\ }\#\_\ \cdots$

多带复制到单带

2. Read the underlined input letters $\in \Gamma^k$ 各带当前字

3. Simulate M by updating the input and the underlining of the head-positions.

通过下标映射模拟动作

4. Repeat 2-3 until M has reached a halting state, M' halt accordingly.

PS: If the update requires overwriting a # symbol, then shift the part $\# \cdots \_$ one position to the right.

A nondeterministic Turing machine M can have several options at every step. It is defined by the 7-tuple $(Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$, with

三大资源

Q finite set of states

Σ finite input alphabet (without "␣")

Γ finite tape alphabet with $\{␣\} \cup \Sigma \subseteq \Gamma$

q_0 start state $\in Q$

q_{accept} accept state $\in Q$

q_{reject} reject state $\in Q$

三大状态

δ the transition function

$$\delta: Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L,R\})$$

转移函数： 一格局有多种前途，在格局的幂集中看是单个元素

类似于公司，集团

考察一个确定性格局演进，用C语言模拟它

$\delta(q1,b) = (q2,c,R)$，

```
M1(Q,  char *pCurr)
 {  if (Q==q1) && (*pCurr ==b)
        { *pCUrr=c; moveRight( ); goto q2;}
 }
```

多CPU 并行
或 广度优先的树
搜索回溯
只要某一分支成功
即可

考察一个不确定性格局演进

$\delta(q1,b) = (q2,c,R) \,||\, (q3,d,L)$

```
M3(Q, *pCurr) {  if (Q==q1) && (*pCurr ==b)
                   return( M1(Q, *pCurr  ) ||
                           M2(Q, *pCurr  ))
                   ..........
              }
```

分时并行调度，每一个走一个时间片

Theorem 3.16 very nondeterministic Turing machine has an equivalent deterministic Turing machine.

因为等价性，以后尽可能用高级工具，多带NTM

Just like k-tape TMs, nondeterministic Turing machines are not more powerful than simple TMs.

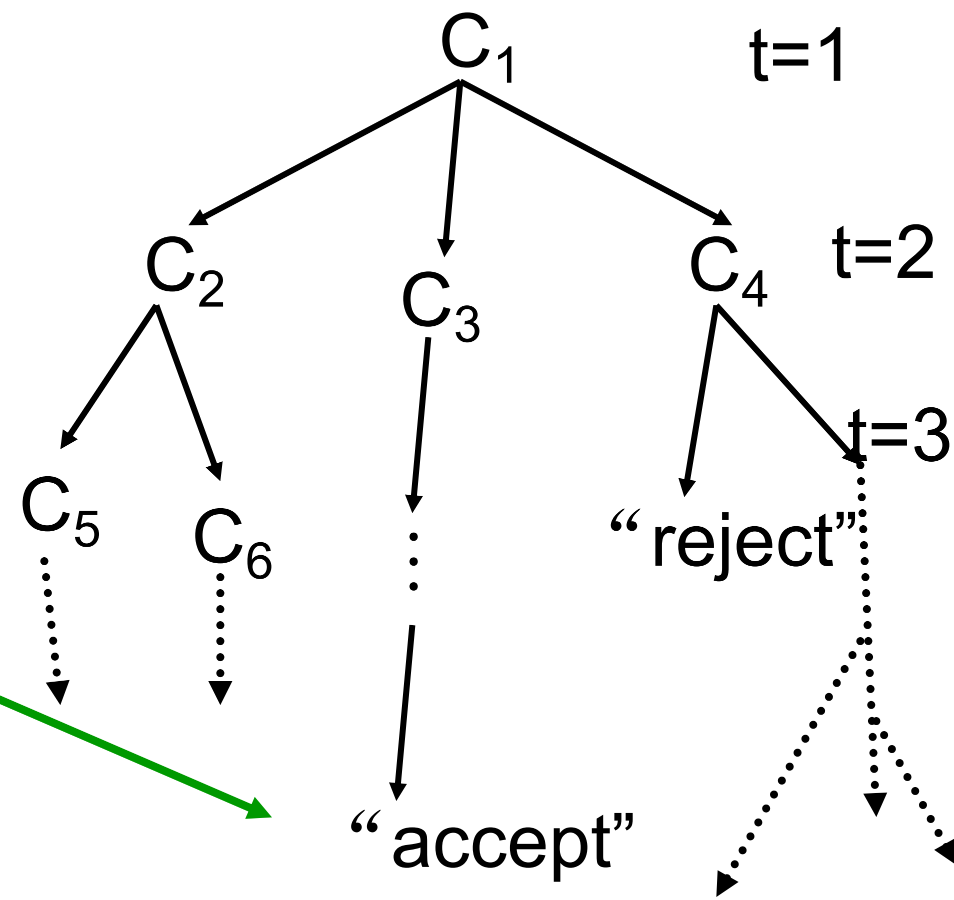
Corollary3.18 A language L is recognizable if and only if some nondeterministic TM recognizes it.

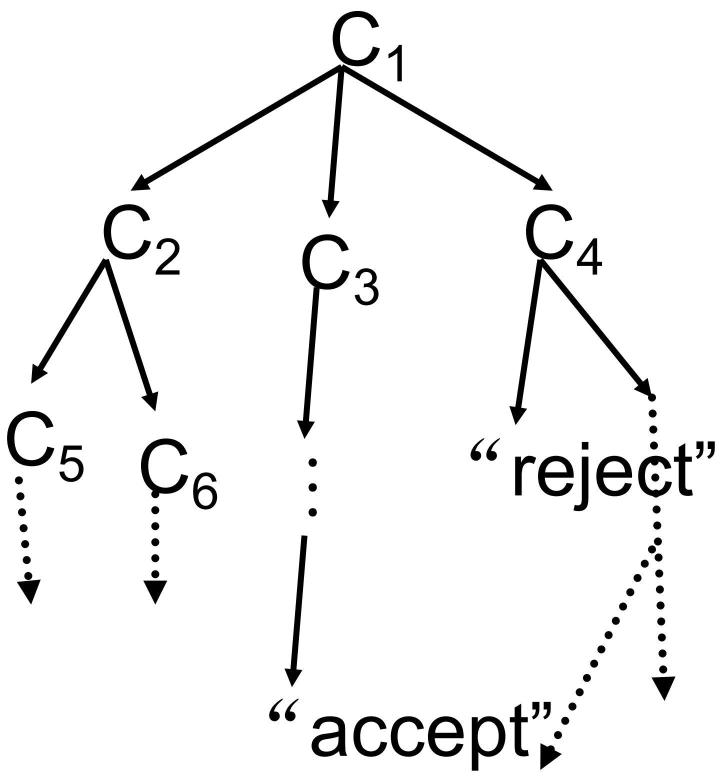
Corollary3.19 A language is decidable if and only if some nondeterministic TM decides it.

The Turing machine model is extremely robust.

Evolution of the NTM represented by a tree of configurations (rather than a single path).
多前途演进是格局树

If there is (at least) one accepting leave, then the TM accepts.
只要一条被接受，就算被接受了，允许多次失败换取一次成功，如彩票





Proof IDEA: **DTM D simulates NTM N;**

1. N's computation on input ω as a tree;

2. A node of the tree corresponds to a configuration.

3. A branch of tree is one branch of the N's computation;

4. TM D search the tree for an accepting configuration, in the breadth-first search rather than depth-first search.

5. Every NTM has an equivalent 3-tape Turing machine, which –in turn– has an equivalent 1-tape Turing machine.

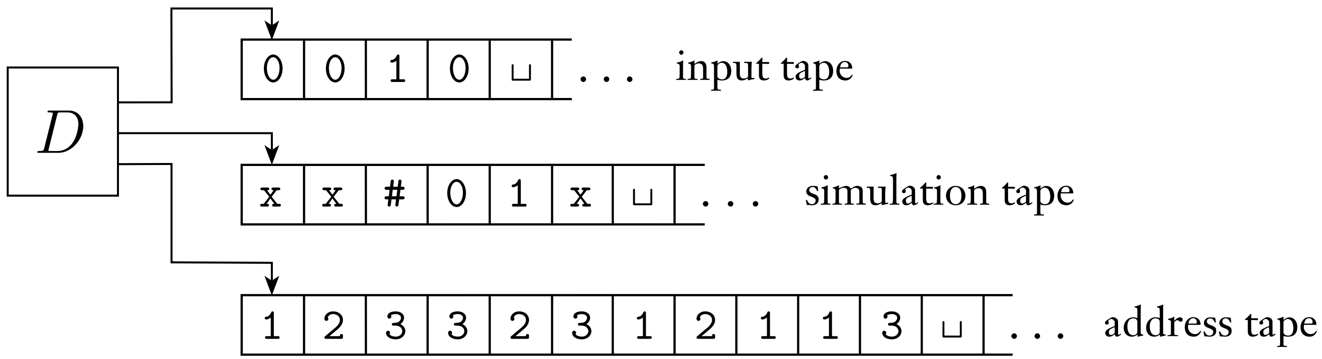


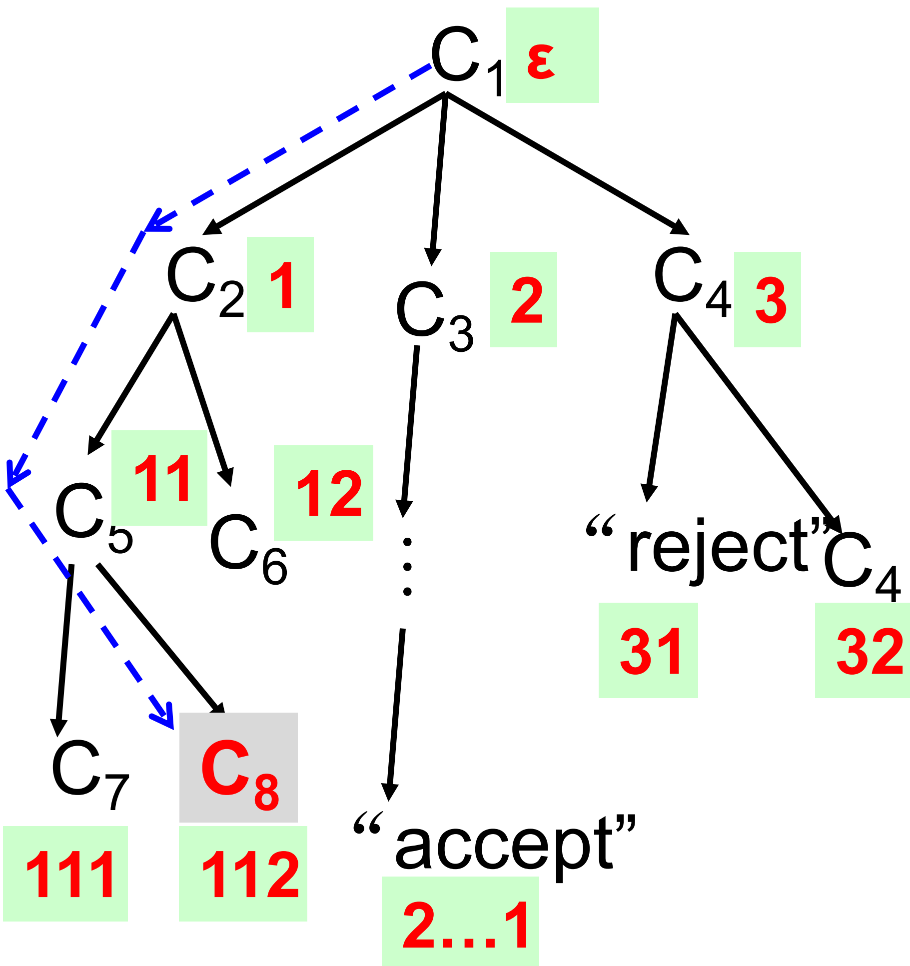
FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

第1带 保存输入 供多次扫描 (不确定，可后悔）

第2带 用于计算 （模拟一个CPU, 一条路线）

第3带 协调调度 不确定过程多个CPU的调度、树搜索与回溯的当前位置，辅助第2带工作。

Variants of TM



Schematic of the node's address

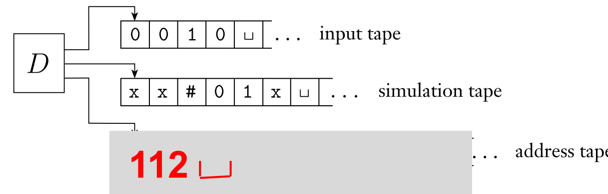


FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

Proof:

1. Initially, tape 1 contains the input ω, tape 2 and 3 are empty;
2. Copy tape 1 to tape 2;
3. Use tape 2 to simulate N with input ω on one branch of its nondeterministic computation……
4. Replace the string on tape 3 with the next string in the string ordering. Simulate the next branch of N's computation by going to stage 2.

■ Enumerator = TM + printer

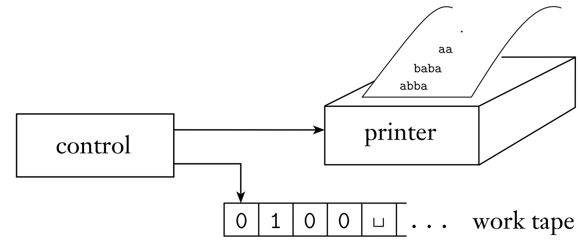


FIGURE 3.20
Schematic of an enumerator

An Enumerator is a 7-tuple $(Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$, where Q,Σ,Γ are all finite sets and

- Q finite set of states
- Σ finite input alphabet (without "_")
- Γ finite tape alphabet with $\{\_\} \cup \Sigma \subseteq \Gamma$
- q_0 start state $\in Q$
- q_{accept} accept state $\in Q$
- q_{reject} reject state $\in Q$

$$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\} \times \Sigma^*$$

$$\delta \quad (q_i,\ a) = (q_j\ ,\ \ b,\ \ L/R,\ \ c\ )$$

read write move print

- Enumerator E starts with a blank input on its work tape.
- Language enumerated by E is the collection of all strings that it eventually print out.

Theorem 3.21 A language is TM-recognizable if and only if it is enumerable. 枚举 = 识别

具体证明见教材p181，主要思想如下：

1. 造M, 它以能被E枚举出 作接受标准;
2. 造E, 把全体字符串按字典顺序输入，它以能被M接受作枚举前提;

Equivalence with Other Models

1. We can consider many other 'reasonable' models of computation: DNA computing, neural networks,quantum computing……
2. Experience teaches us that every such model can be simulated by a Turing machine. So, they are equivalent (recognize the same language).
3. Church-Turing Thesis: 现在提出的计算模型都可用图灵机模拟（*simulate*）。

*The intuitive notion of **computing and algorithms** is captured by the **Turing machine** model.*

The Church-Turing Thesis

The Church-Turing thesis marks the end of a long sequence of developments that concern the notions of "way-of-calculating", "procedure", "solving", "algorithm".

意义：C-T论题为以前不精确的议论给出了数学模型

Goes back to Euclid's GCD algorithm (300 BC).

For a long time, this was an implicit notion that defied proper analysis.

Importance of the Church-Turing Thesis

什么是算法：现在可以说：图灵机就是算法的数学模型

The Church-Turing thesis marks the end of a long sequence of developments that concern the notions of “way-of-calculating”, “procedure”, “solving”, “algorithm”.

辗转除法求 最大公约数

Goes back to Euclid’s GCD algorithm (300 BC).

For a long time, this was an implicit notion that defied proper analysis.

Hilbert's 10th Problem 希尔伯特第 10 问题

In 1900, David Hilbert (1862–1943) proposed his *Mathematical Problems* (23 of them).

The Hilbert's 10th problem is: **Determination of the solvability of a Diophantine equation.**

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: *To devise a process according to which it can be* determined by a finite number of operations *whether the equation is solvable in rational integers.*

是否有算法（有限步）判定 整系数不定方程有有理数解）

(Un)solving Hilbert's 10th

Hilbert's "*…a process according to which it can be determined by a finite number of operations…*" needed to be defined in a proper way.

提出为有限过程形式化描述的需求

指数方程问题已被证明不可判定

第10问题已经被证明不可判定,1970

The impossibility of such a process for exponential equations was shown by Davis,Putnam and Robinson.

Matijasevič proved that Hilbert's 10th problem is unsolvable in 1970.

Hilbert's 10th Problem

整系数不定方程求解算法　（学会形式化描述问题）：

Let $P(x_1,\ldots,x_k)$ be a polynomial in k variables with integral coefficients. Does P have an integral root $(x_1,\ldots,x_k) \in Z^k$?

Example: $P(x,y,z) = 6x^3yz + 3xy^2 - x^3 - 10$ has integral root $(x,y,z) = (5,3,0)$. 有解

有解

Other example: $P(x,y) = 21x^2 - 81xy + 1$ does not have an integral root. 无解

无解

解决希尔伯特第10问题的努力

D = {p | p是有整数根的多项式} （多个变元多项式）

希尔伯特第10问题是问：集合D是否可判定？

答案：否

一个变元的多项式，如4x3-2x^2+x-7。

问题得到了简化，考虑单个变元多项式

D1 = {p | p是有整数根的x的多项式} M1 =“输入是关于变元x的一个多项式p：

```
while (1)
{   x=0
  if (( P(x) =0)|| P(-x) =0)) return Yes ;
  else x++;
}
```

有根， 会返回；无根，可能无限循环。M1是识别器，不是判定器。

Hilbert's 10th Problem

单个变元多项式算法的改进：

D1 = {p | p是有整数根的x的多项式} 识别器图灵机M1：

项数

最大系数

首项系数

有根，则返回；如无根，可能无限循环。M1是判定器。

$$UpBound = K\left(C_{Max}/C_1\right)$$

```
while (X<UpBound)
{   x=0
  if (( P(x) =0)|| P(-x) =0)) return Yes ;
 else x++;}
```

Hilbert's 10th Problem

类似的方法能用到多元多项式算法吗？：

UpBound=？ ？

Matijasevic 1970. 证明
对多元多项式，这个上界不可能计算

即判定proved that Hilbert's 10th problem is unsolvable

```
while (x<UpBound)
{    x=0
  if (( P(x) =0)|| P(-x) =0)) return Yes ;
  else x++;
}
```

Church-Turing Thesis

- 算法的定义

算法 ≅ 处处停机图灵机
= λ-演算 （Alonzo Church提出）
= 0型文法

- 丘奇-图灵论题（C-T Thesis）

算法的直觉概念=图灵机算法

总之，算法可以用图灵机定义，图灵机是定义算法的精确模型。

Describing TM Programs一些约定

Three Levels of Describing algorithms:

- formal (state diagrams, CFGs, et cetera) 状态图,文法
  - ✧ Details of states, transition function, and so on.
- implementation (pseudo-Pascal) （**PASCAL伪码**）
  - ✧ Operation of the head and the tape.
- high-level (coherent and clear English) 英语
  - ✧ Ignoring the implementation details.

Describing TM Programs一些约定

Three Levels of Describing algorithms:

- formal (state diagrams, CFGs, et cetera) 状态图,文法
- implementation (pseudo-Pascal) 实现（伪码）
- high-level (coherent and clear English) 高级描述

输入输出：串

Describing input / output format:

TMs allow only strings $\in\Sigma^*$ as input/output.

If our X and Y are of another form (graph, Turing machine, polynomial), then we use <X,Y> to denote 'some kind of **encoding** $\in\Sigma^*$' 编码

如问题描述不是串，编码成为串

TONGJI UNIVERSITY

Decidability

We are now ready to tackle the question:

What can computers do and what not?

问题不精确
不容易直接回答

By Church-Turing thesis 转化为下列问题

Which languages are TM-decidable, Turing-recognizable, or neither?

那些是图灵可识别，可判定或都不是?
问题精确　容易多了

Review

1. 自动机与语言；
2. 图灵机的定义：示意图、状态转移图、形式化定义、**high-level**描述；
3. 图灵机的变种：多带图灵机、非确定性图灵机、枚举器；
4. 算法与丘奇-图灵论题；
5. 算法描述的三个层次：**formal**，**implementation**，**high-level.**

Thank You !