中国科学院大学
University of Chinese Academy of Sciences

计算机科学导论

期末复习

徐志伟
中科院计算所 中国科学院大学
zxu@ict.ac.cn

0. 同学们已经学到了不少知识

美国科学院与工程院

- “计算机科学是研究计算机以及它们能干什么的一门学科。它研究**抽象计算机**的能力与局限，**真实计算机**的构造与特征，以及用于求解问题的无数**计算机应用**。”
  - 计算机科学涉及**符号**及其操作；
  - 关注多种**抽象**概念的创造和操作；
  - 创造并研究**算法**；
  - 创造各种**人工结构**，尤其是不受物理定律限制的结构；
  - 利用并应对**指数增长**；
  - 探索计算能力的**基本极限**；
  - 关注与人类**智能**相关的复杂的、分析

图灵机 自动机

冯诺依曼计算机
笔记本电脑

班级排序
信息隐藏GO程序
个人作品

从电路到算法的
各种抽象

布尔逻辑
命题逻辑
谓词逻辑

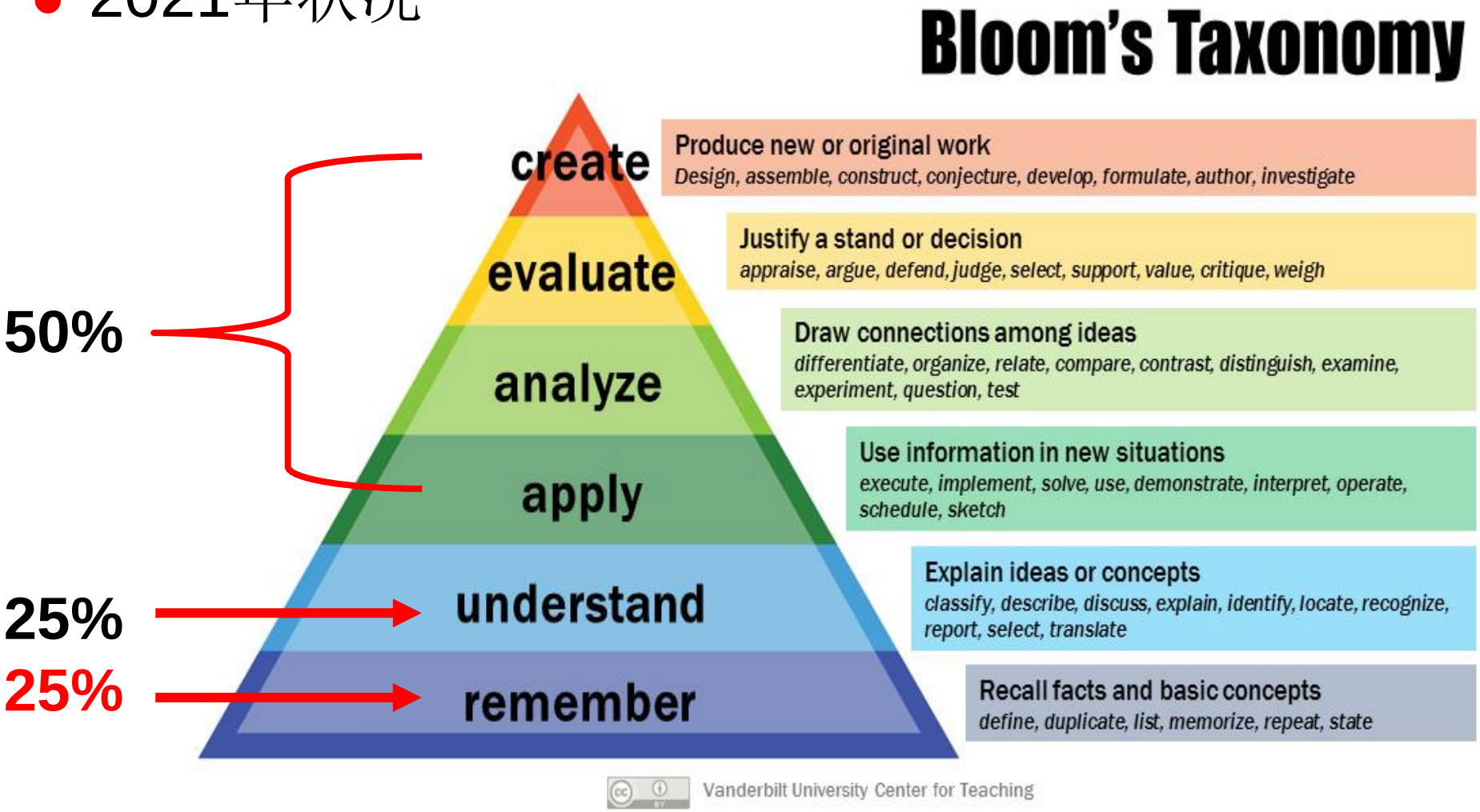
各种建模

斐波那契递归程序
P与NP

图灵机不可计算问题
哥德尔不完备定理

当代教育学理论指导：布鲁姆教学目标分类法（2001修订版）

- 2021年状况



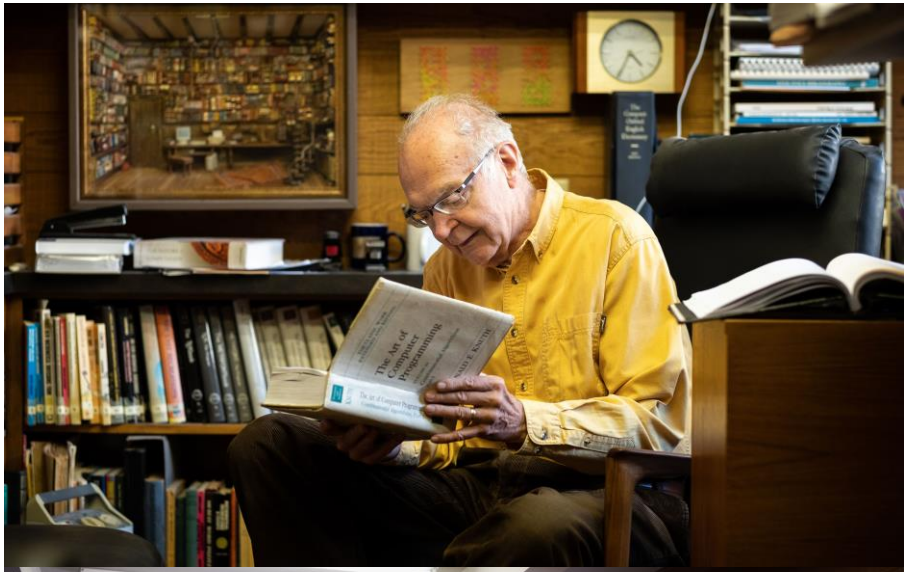
https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/

高德纳忠告
是否掌握知识的终极测试

“The ultimate test of whether I **understand something** is if I can **explain it to a computer**. I can say something to you and you’ll nod your head, but I’m not sure that I explained it well. But the computer doesn’t nod its head. It repeats back exactly what I tell it. In most of life, you can bluff, but not with computers.”

计算机科学、
数理化天地生、
工程、经管、艺术

Donald Knuth, Feb 2020



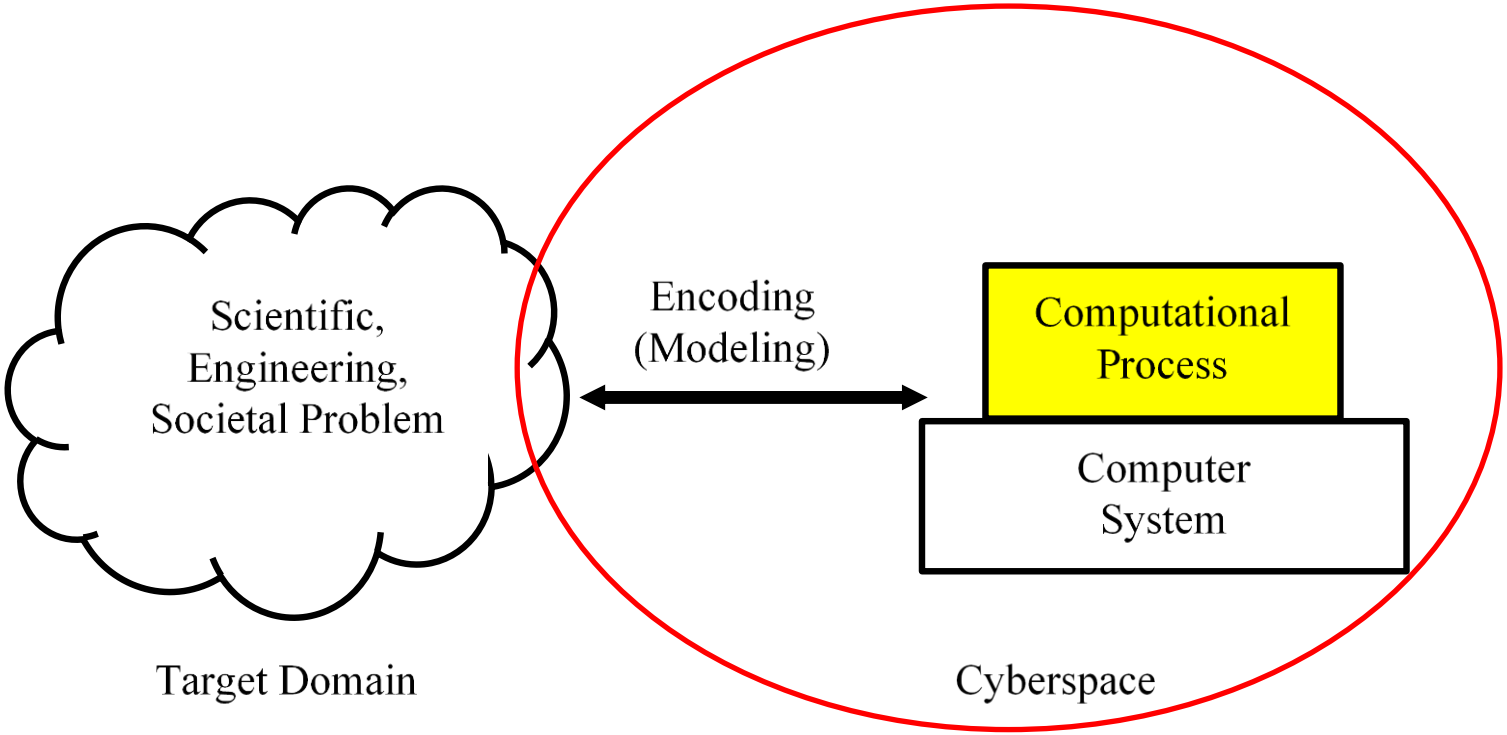
Photos by Vivian Cromwell

教科书**75-76**页，有关高德纳

Susan D'Agostino. The Computer Scientist Who Can’t Stop Telling Stories. Quanta Magazine. April 16, 2020. https://www.quantamagazine.org/computer-scientist-donald-knuth-cant-stop-telling-stories-20200416.

1.1 计算机科学绪论

- 什么是计算机科学?
  - 计算机科学是研究**计算过程，有效解决问题**的科学
  - 计算过程是**信息变换**过程
    - 即**通过操作数字符号变换信息的过程。**



计算机科学影响广泛
为什么这么广泛？

Why Computer Science Permeates Our Civilization

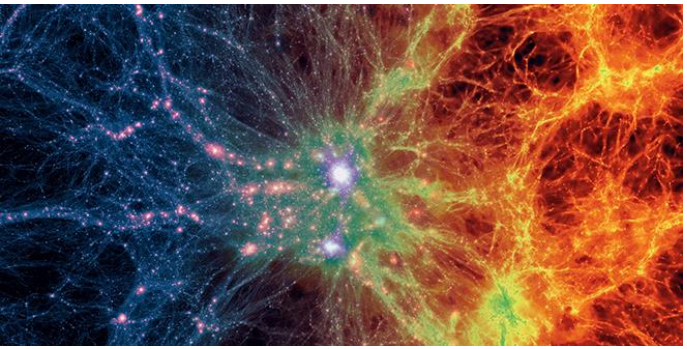
Chomsky's digital infinity principle: A finite set of digital symbols can be combined to produce many domain languages with infinite expressions.

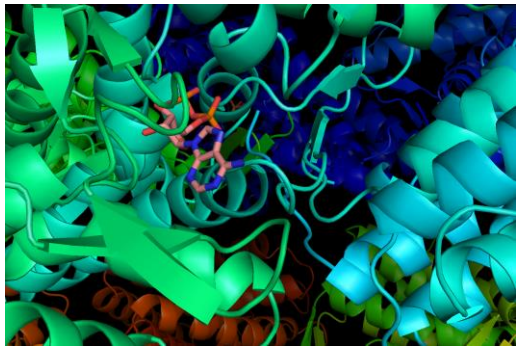
Karp's computational lens thesis: Many processes in Nature and human Society are also computational processes. Nature computes. Society computes. We can understand Nature and Society better through the computational lens.

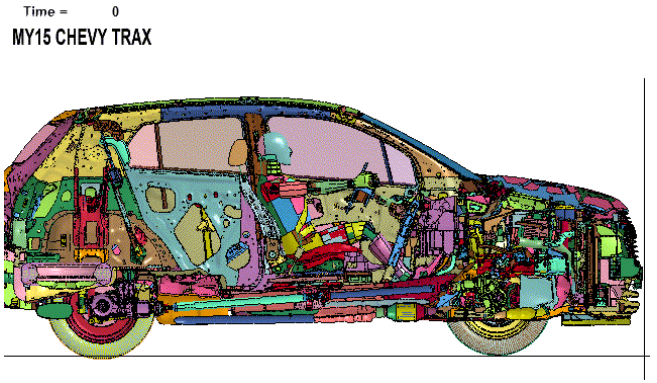
Babayan's gold metaphor: Computing speed is like gold, a hard currency that can be exchanged for anything. This assertion was made in the HPC-Asia 2000 Conference in Beijing by Boris Babayan, a Russian computer scientist.

Boutang's bees metaphor: ICT is like bees, producing two type of outputs. The indirect output (pollination) of bees has economic value that is orders of magnitude larger than the value of the direct output (honey). Similarly, the value of digital economy (pollination) is much larger than that of the ICT market (honey).

为什么计算机科学渗透广而深？







科学、技术、经济、社会的各种问题和过程

各领域的专业表达

都可以用该领域的语言表示

数字无穷性假说
+
计算透镜假说

有穷数字符号的组合可表达无穷语言
+
Nature computes.
Society computes.

计算问题和计算过程

给地球做CT

计算思维的外部特征和内部特征

- 外部特征：强调比特精准、抽象构造、自动执行
  - ABC特征：Automatic Execution, Bit Accuracy, Constructive Abstraction
- 内部特征： Acu-Exams-CP

(A): Automatically execution	**自动执行**。计算机能够自动执行由离散步骤组成的计算过程。
(C): Correctness	**正确性**。计算机求解问题的正确性往往可以精确地定义并分析。
(U): Universality	**通用性**。计算机能够求解任意可计算问题。
(E): Effectiveness	**构造性**。人们能够构造出聪明的方法让计算机有效地解决问题。
(X): Complexity	**复杂度**。这些聪明的方法（算法）具备时间/空间复杂度。
(A): Abstraction	**抽象化**。少数精心构造的计算抽象可产生万千应用系统。
(M): Modularity	**模块化**。多个模块有规律地组合成为计算系统。
(S): Seamless Transition	**无缝衔接**。计算过程在计算系统中流畅地执行。
(C): Connectivity	**连接性**。很多问题涉及用户/数据/算法的连接体，而非单体。
(P): Protocol Stack	**协议栈**。连接体的节点之间通过协议栈通信交互。

计算思维的外部特征和内部特征

- 外部特征：强调比特精准、抽象构造、自动执行
  - ABC特征：Automatic Execution, Bit Accuracy, Constructive Abstraction
- 内部特征： Acu-Exams-CP

逻辑思维：正确

Logic thinking makes computational processes correct

算法思维：巧妙

Algorithmic thinking makes computational processes smart

系统思维：实用

Systems thinking makes computational processes practical

They are a symphony

求Fibonacci number F(10)

- 手算

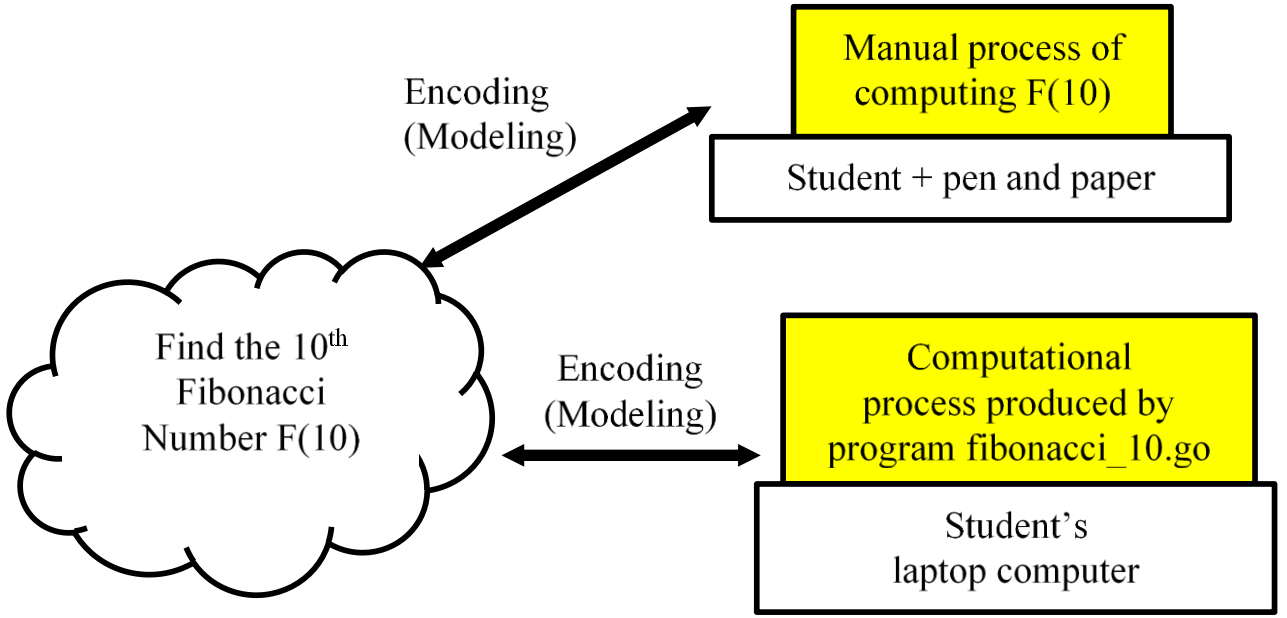
F(2)=F(1)+F(0)=1+0=1,
F(3)=F(2)+F(1)=1+1=2,
F(4)=F(3)+F(2)=2+1=3,
F(5)=F(4)+F(3)=3+2=5,
F(6)=F(5)+F(4)=5+3=8,
F(7)=F(6)+F(5)=8+5=13,
F(8)=F(7)+F(6)=13+8=21,
F(9)=F(8)+F(7)=21+13=34,
F(10)=F(9)+F(8)=34+21=55.

求F(50)，F(5000)， F(5,000,000,000)时，分别会发生什么？

它们如何反映了ABC与Acu-Exams？

- 计算机算

```
package main
import "fmt"
func main() {
   fmt.Println("F(10)=", fibonacci(10))
}
func fibonacci(n int) int {
   if n == 0 || n == 1 {
      return n
   }
   return fibonacci(n-1)+fibonacci(n-2)
}
```



五个斐波那契算法与程序的比较

- T(n)是计算F(n)程序的执行时间in seconds
- 注意：
  - 头三个程序并不实用，只能计算到F(92)
  - 第四个程序计算第十亿个斐波那契数时耗时太长
    - 为什么这么长？48小时 >> 102*200=20400秒（不到7小时）
  - 第四、五个程序需要系统提供更多支持
- **系统思维使得计算过程实用（practical）！**

Big-O忽略了常数。
复杂度分析忽略了大数。
F(1亿+1)+F(1亿+2)远比
F(1)+F(2)耗时
不能忽略字长
不能忽略整数-字符转换时间

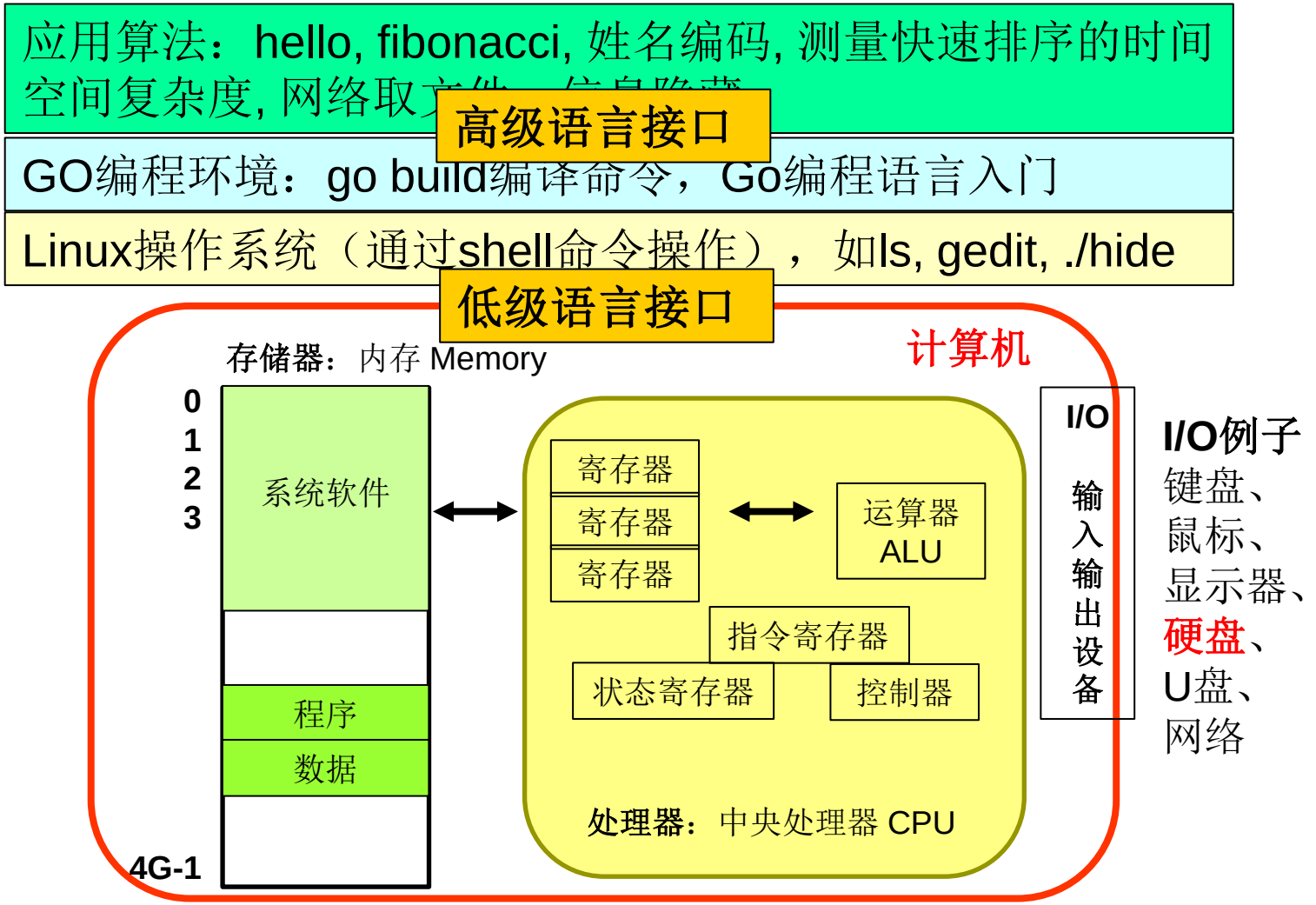
算法与程序	普通递归 fibonacci_recursive.go	动态规划-递归 fibonacci_recursive_dp.go	动态规划-迭代 fibonacci_dp.go	大数动态规划 fib_dp.go	最优化版本 fib.go
时间复杂度	$O(2^n)$	O(n)	O(n)	O(n)	O(logn)
空间复杂度	O(n)	O(n)	O(n)	O(1)	O(1)
T(50)	725s	0.059s	0.057s	0.019s	0.000012s
T(500)	出错	出错	出错	0.026	0.000022s
T(5000000)				102	4.129663s
T(1000000000)				两天后中断	187160.0s

1.2 在冯诺依曼机上操作数字符号

涉及软件和硬件

软件

硬件

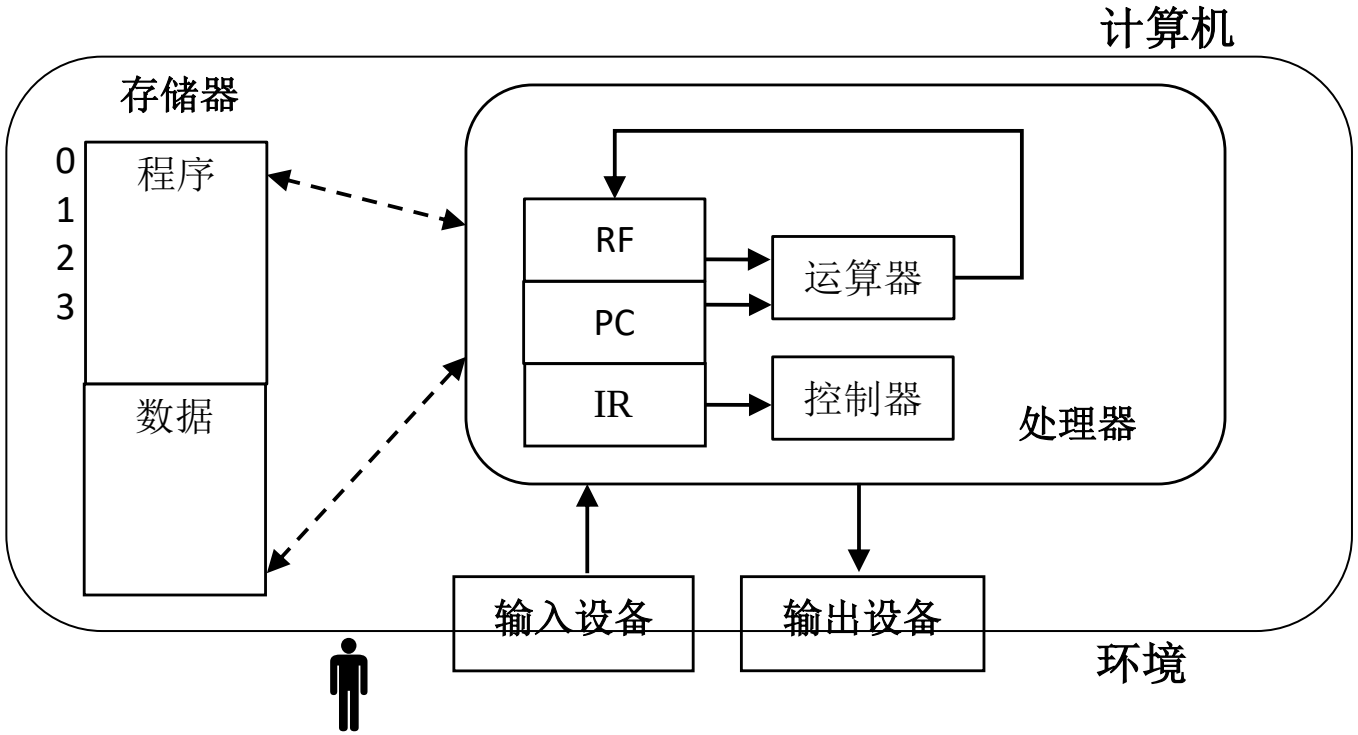


当代计算机基本组成：**冯诺依曼模型简介**

存储程序计算机（冯诺依曼模型）

- 二进制数据及其算术逻辑操作
- 计算机 = 处理器 + 存储器 + 输入输出设备
  - 处理器 = 运算器+控制器+寄存器
  - 【当代处理器的核心是指令流水线】
- 存储程序计算机（stored program computer）
- 串行执行

RF: register file
PC: program counter
IR: instruction register



二进制表示的通用性

- $(110.101)_2=1\times2^2+1\times2^1+1\times2^{-1}+1\times2^{-3}=(6.625)_{10}$
- 6.625的二进制表示是什么？
- 课程讨论了如下数据类型的二进制表示和操作
  - 同学们通过GO编程练习动手掌握
    - 比特
    - ASCII字符
    - 自然数
    - 整数
    - ……
    - 数组、切片
    - bmp图像文件

符号有窍门
二进制补码例子：-127到127的整数加法

- 直截了当的表示
  - 需要8比特，2^7=128，以及1比特表示正负
  - 63 = 00111111, 64 = 01000000, -63=11000001, -64=11000000
  - (-63) = 10111111，(-64) = 11000000
  - 63 + 64 = 00111111 + 01000000 = 01111111 = 127
  - (-63) + (-64) = 10111111 + 11000000 = 11111111 = (-127)
  - 63 + (-63) = 00111111 + 10111111 = 11111110 = (-126) 错！
- 补码表示
  - 负数：绝对值的逐位取反，然后加00000001，即从最低一位加1
  - (-63) = 00111111逐位取反 + 00000001 = 11000000 + 00000001 = 11000001
  - 63 + (-63) = 00111111 + 11000001 = 00000000 = 0 正确！

补码加法更多细节

- 给定数的绝对值二进制表示。如，|+63|=|-63| = 00111111
- 正数（+63）：绝对值表示。 00111111
- 负数（-63）：绝对值逐位取反，然后加00000001。11000000+00000001=11000001

- 63+63 = 00111111+00111111= 100000000= 01111110（无溢出）

  00111111

  01111110 = 126

- 63+(-63) = 00111111+11000001= 100000000= 00000000 （忽略最高位溢出的1）

  11000001

  100000000 = 00000000 = 0

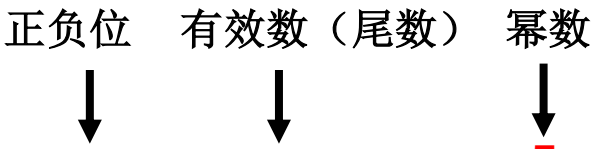
- -63+(-63)= 11000001+11000001= 110000010= 10000010 （忽略最高位溢出的1）

  11000001

  110000010 = 10000010 = -126

如何表示实数：浮点数概念

- 本课程编程练习中不涉及
- 特定的有穷数字符号组合不能精确表达无穷数
- 办法：近似表达
  - 例如，圆周率π=3.14159265…，假设6位十进制精度



  - $3.14159 = + 314159 \times 10^{-5}$

- 一个浮点数可由两个整数（定点数）表示

ASCII字符

ASCII码=0$D_6D_5D_4D_3D_2D_1D_0$

ASCII字符编码 “Xu Zhi Wei” = [88,117,32,90,104,105,32,87,101,105]

$D_3D_2D_1D_0$ \ $D_6D_5D_4$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	”	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

SP= 00100000_2 $=32_{10}$

空格，即SP是ASCII字符

对应的数值
00100000
或
32
称为ASCII码

最高位可假定是0
有时用来作奇偶校验位

加号+的ASCII码？

00101011_2
43_{10}

Esc
00011011_2
27_{10}

空字符
00000000_2
0_{10}

字符 ‘0’
00110000_2
48_{10}

name_to_number-0.go vs. name_to_number.go

- 占位符%d

```
package main
import "fmt"
func main() {
	var name string = "Xu Zhi Wei"
	sum := 0
	for i := 0; i < len(name); i++ {
		sum = sum + int(name[i])
	}
	fmt.Printf("%d\n", sum)
}
```

```
> ./name_to_number-0
> 861
>
```

```
1  package main
2  import "fmt"
3  func main() {
4  		var name string = "Xu Zhi Wei"
5  		var sum int = 0
6  		var i int
7  		for i = 0; i < len(name); i++ {
8  				sum = sum + int(name[i])
9  		}
10 		var sum_bytes [5]byte
11 		var j int
12 		for j = len(sum_bytes) - 1; sum != 0; j-- {
13 				sum_bytes[j] = byte(sum % 10) + '0'
14 				sum = sum / 10
15 		}
16 		var k int
17 		for k = j + 1; k < len(sum_bytes); k++ {
18 				fmt.Printf("%c", sum_bytes[k])
19 		}
20 		fmt.Printf(\n)
21 }
```

```
> ./name_to_number
> 861
>
```

难点：用**%c**占位符 实现 **%d**占位符

位值表示的重要性

- Find a person's age using Roman numerals and decimal numerals
  - **A person was born in year MCMLIV. What's his age in year MMXXI?**
  - **MMXXI – MCMLIV = ?** 2021 – 1954 = 67
    - Value of MCMLIV is the sum of digit values. E.g., MCMLIV=M+CM+L+IV=1000+900+50+4=1954

Roman	M	D	C	L	X	V	I	IV	IX	XL	XC	CD	CM
Decimal	1000	500	100	50	10	5	1	4	9	40	90	400	900

罗马表示
vs.
十进制位值表示

- Decimal number system uses a **positional notation**.
  - Consider both digit values and positional weights
- Example: $a = a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}a_{-4} = 2021.1954$
- The number *a* has eight digits
  - Two digits have identical symbol 2.
  - But represent different values:
    - two thousands and two tens.
  - Value of $a = \sum_{i=-4}^{3} a_i \times \mathbf{10}^i$.
- 可以用无理数为底吗？

用有限位表示无理数：100= 4 $\tau = (1.618\ldots)^2$ =2.618… 3

$\tau = (1+\sqrt{5})/2 = 1.618\ldots$ F(6)F(5)F(4)F(3)F(2)

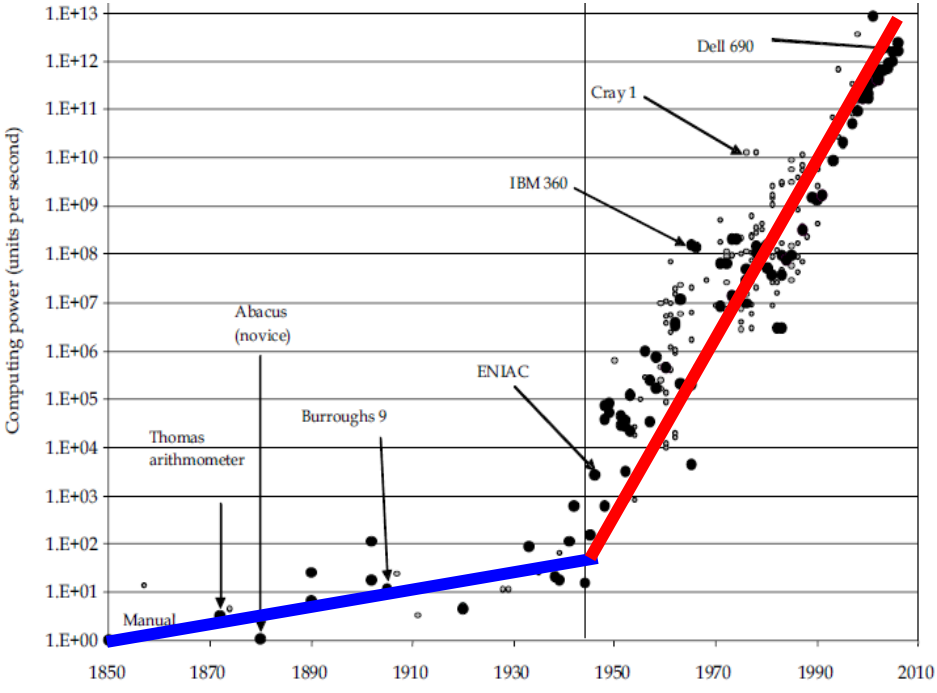
Decimal	Hexadecimal	Binary	The τ Number System	FNS
$10^1 10^0$	16^0	$2^3 2^2 2^1 2^0$	$\tau^5\tau^4\tau^3\tau^2\tau^1\tau^0\tau^{-1}\tau^{-2}\tau^{-3}\tau^{-4}\tau^{-5}\tau^{-6}$	8 5 3 2 1
0	0	0000	0	00000
1	1	0001	1	00001
2	2	0010	10.01	00010
3	3	0011	100.01	00100
4	4	0100	101.01	00101
5	5	0101	1000.1001	01000
6	6	0110	1010.0001	01001
7	7	0111	10000.0001	01010
8	8	1000	10001.0001	10000
9	9	0001	10010.0101	10001
10	A	1010	10100.0101	10010
11	B	1011	10101.0101	10100
12	C	1100	100000. 101001	10101
13	D	1101	100010.001001	11000
14	E	1110	100100.110110	11001
15	F	1111	100101.001001	11010

1.3 当代计算机科学的三个基本追求

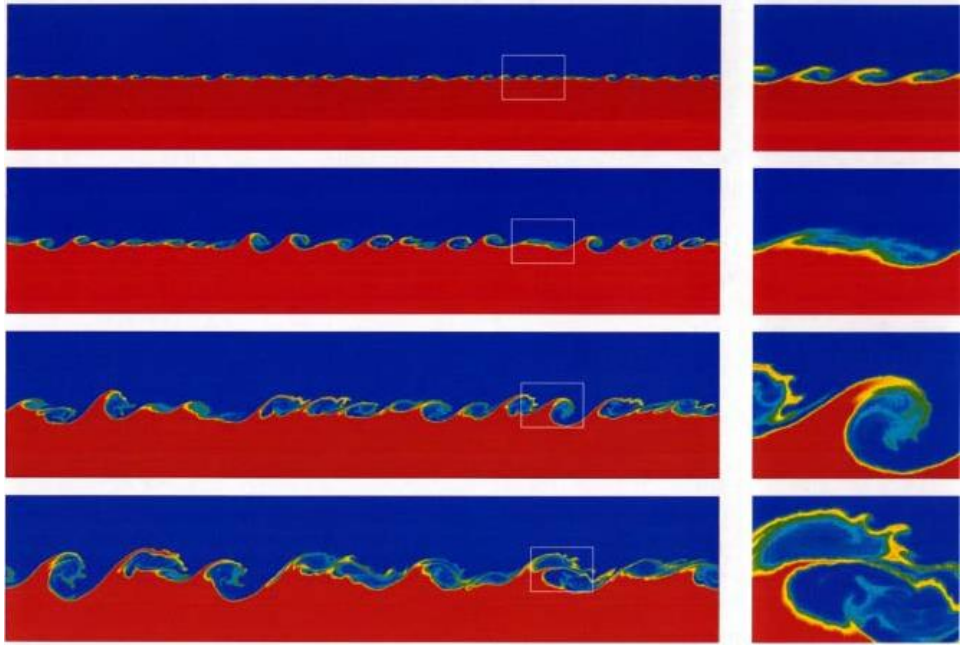
- **巴比奇问题**：如何构造有效的计算系统？
  - Charles Babbage，19世纪剑桥大学教授
  - 提出了差分机和分析机概念，是计算机系统先驱
- **布什问题**：如何有效地使用计算系统，人-计算机-信息如何互联？
  - Vannevar Bush，20世纪麻省理工学院教授
  - 发明了微分分析机，提出了Memex个人计算机
- **图灵问题**：如何用计算机解题、最终产生智能？
  - Alan Turing，20世纪英国计算机科学家艾伦·图灵
  - 提出了图灵机与图灵测试，是计算机科学理论的奠基人与人工智能先驱

1.4 计算机科学的三个奇妙
指数之妙
Wonder of Exponentiation

- 问题的运算量随问题规模指数增长：$C=O(2^n)$
  - 蛋白质折叠的运算量 ≈ 3^{300} ≈ 10^{143}；问题规模n=300
    - 挑战：将运算量降为 1.2^n、甚至n^k，k是一个小的常数
- 计算速度随时间指数增长
  - Nordhaus's law：
    1945-2006期间，
    单台计算机的
    计算速度平均
    每年增长50%
  - 为什么出现拐点？



模拟之妙
Wonder of Simulation

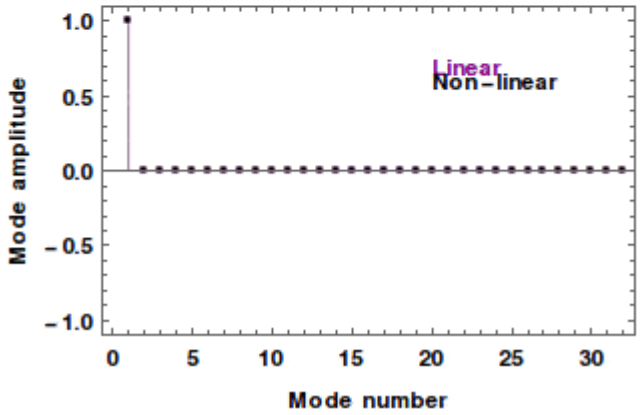


- 汽车碰撞模拟
- 核武器模拟
- 从第一原理重现宏观现象
  - 科学家们常常需要知道一些宏观现象是如何产生的。但尚未总结出刻画现象的方程。
  - 常用的科学方法：
    - 从物理学的第一原理（如牛顿力学）出发，通过计算机模拟，生成宏观现象。使得人们能够看见原来看不见的物理过程。
  - 美国能源部的科学家使用超级计算机模拟了90亿个原子的运动，产生出Kelvin-Helmholtz不稳定性的宏观现象。
- Atoms in the Surf视频讲解

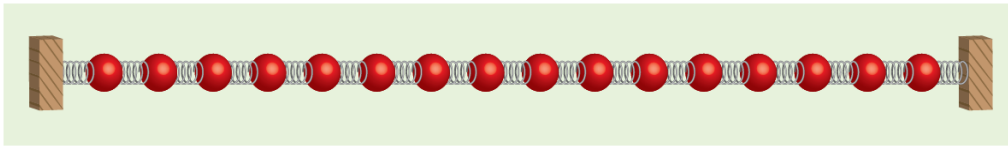
1953年，费米等发明计算机模拟

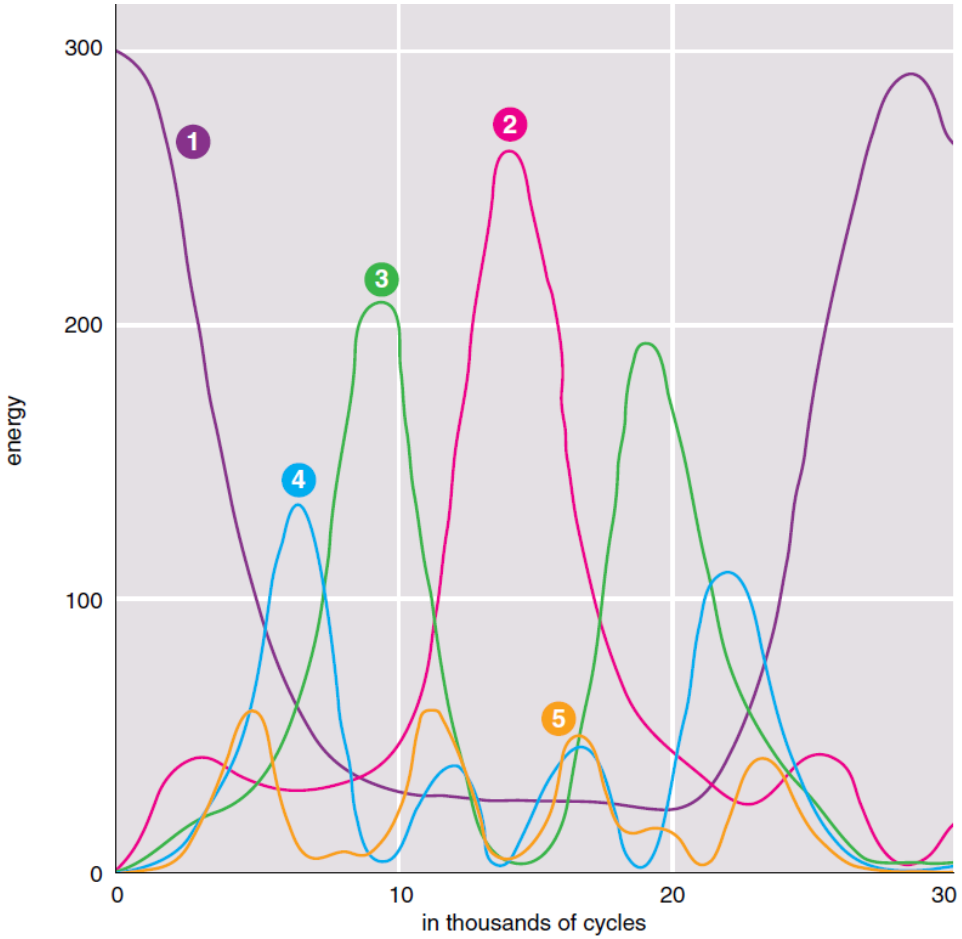
- 除了理论、实验之外的第三种科学研究方法
- Fermi–Pasta–Ulam悖论
  - FPU Recurrence

Onorato, *et al*, PNAS 2015
提供了部分解



en.wikipedia.org/wiki/Fermi–Pasta–Ulam–Tsingou_problem





Porter, Mason A., et al. "Fermi, Pasta, Ulam and the Birth of Experimental Mathematics: A numerical experiment that Enrico Fermi, John Pasta, and Stanislaw Ulam reported 54 years ago continues to inspire discovery." *American Scientist* 97.3 (2009): 214-221.

虚拟之妙
Wonder of Cyberspace

- 计算世界（cyberspace）是人创造的
  - 可在虚拟世界中创造出与现实世界平行、甚至现实世界没有的东西
    - 10亿本书的书店
    - 1亿家店商的购物中心
    - 1000个队员的足球赛
    - Earth-diameter virtual telescope 地球规模望远镜
- 虚拟性使得一切全在设计者和创造者的掌控之中
  - 吸引很多年轻人加入计算机科学领域、成为创新者

2. 逻辑思维

- 比特精准的最基本体现
- 关注计算过程的比特精准的正确性与通用性
- 课程讨论了
  - 布尔逻辑
  - 图灵机
  - 图灵机的通用性与局限
  - 哥德尔不完备定理
- 说明每个概念，并给出具体例子

逻辑思维要点

- **比特精准地定义、建模、推导和证明逻辑命题**
  - 能自己想明白
  - 能和别人说明白

- 例子：存在无穷多个素数。
  - 证明方法：假如只有有限个素数，...，得出矛盾
  - 使用谓词逻辑，我们有：

$$\forall n, \exists m, [(m > n) \wedge (\text{Prime}(m))]$$

$$= \neg\neg(\forall n, \exists m, [(m > n) \wedge (\text{Prime}(m))])$$

$$= \neg(\exists n, \forall m, [(m > n) \rightarrow \neg(\text{Prime}(m))])$$

实例：全等三角形一定相似

- 引入问题：用“若x则y”的形式，写出“全等三角形一定相似”的四种命题，并判断它们的真假。
- 解答
  - 原命题：若是全等三角形，则它们一定是相似三角形。 $x \rightarrow y$
  - 逆命题：若是相似三角形，则它们一定是全等三角形 $y \rightarrow x$
  - 否命题：若不是全等三角形，则它们一定不是相似三角形 $(\neg x) \rightarrow (\neg y)$
  - 逆否命题：若不是相似三角形，则它们一定不是全等三角形 $(\neg y) \rightarrow (\neg x)$

- 原命题和逆否命题为等价命题
- 否命题和逆命题为等价命题
- 为什么？根据布尔逻辑基本性质可以推导出，不需要几何知识
  - $x \rightarrow y = \neg x \vee y$
  - $(\neg y) \rightarrow (\neg x) = (\neg(\neg y)) \vee (\neg x) = y \vee \neg x = \neg x \vee y$

可自动执行！

2.1 布尔逻辑

- 命题逻辑、布尔函数、组合电路
  - 公理系统（从代数角度理解）
    - 元素：常数（0,1）、变量；布尔表达式
    - 算子：基本操作、逻辑连接符
      - 与：$x \wedge y$，$x \bullet y$；或：$x \vee y$，$x+y$；非：$\neg x$，$\overline{x}$；蕴含；异或
    - 公理（与性质）
    - 推理规则
  - 用真值表辅助
- 谓词逻辑
  - 比命题逻辑多了断言和量词（**全称量词∀**和**存在量词∃**）
    - 断言是会传回“真”或“假”的函数
      - 任何自然数，要么它是偶数，要么它加1后为偶数。
      - $\forall n\ [\mathrm{Even}(n) \vee \mathrm{Even}(n+1)]$

布尔逻辑的常量、变量、连接符

- 布尔常量：真 T (true)，假 F (false)
- 合取，与 $\wedge$ (conjunction, and)
  - $x \wedge y = 1$ (T) *iff* $x = y = 1$ (T)
- 析取，或 $\vee$ (disjunction, or)
  - $x \vee y = 0$ (F) *iff* $x = y = 0$ (F)
- 非 $\neg$ (negation, not)
  - $\neg x = 1$ (T) *iff* $x = 0$ (F)
- 蕴含 $\rightarrow$ (implication)
  - $(x \rightarrow y) = 1$ (T) *iff* $x = 0$ (F) or $y = 1$ (T)
- 异或 $\oplus$ (exclusive or)
  - $x \oplus y = 1$ (T) *iff* $x \neq y$

今天下雨。$a^2 \geq 0$。

山无陵，江水为竭，冬雷震震，夏雨雪，天地合，乃敢与君绝。

布尔函数

- 系统思维：组合电路能实现任意布尔函数吗？
  - 组合电路：由与、或、非门电路组合而成的逻辑电路

- n-输入-1输出的布尔函数是数学函数 $f:\{0,1\}^n \to \{0,1\}$
  - 如，$f(x_1, x_2, \cdots, x_n) = (\vee_{i=1}^{n-1} x_i) \oplus x_n$
- 两个布尔函数相同：有相同的真值表
  - 类似：$f(x,y) = x^2 - y^2$和$g(x,y) = (x+y)(x-y)$是相同的多项式
- 一个变量x的布尔函数
  - 一共四个

x	y
0	**0**
1	**0**

$y = 0$

x	y
0	**1**
1	**1**

$y = 1$

x	y
0	**0**
1	**1**

$y = x$

x	y
0	**1**
1	**0**

$y = \overline{x}$

任意布尔函数有范式（唯一性）

- 合取范式(conjunctive normal form, CNF)
  - $f(x_1,\ldots,x_n)= Q_1 \wedge Q_2 \wedge Q_3 \ldots \wedge Q_m$
  - 其中: $Q_i = l_1 \vee l_2 \vee \ldots \vee l_n$，　$l_j = x_j$ 或 $\neg x_j$
- 析取范式(disjunctive normal form, DNF)
  - $f(x_1,\ldots,x_n)= Q_1 \vee Q_2 \vee Q_3 \ldots \vee Q_m$
  - 其中: $Q_i = l_1 \wedge l_2 \wedge \ldots \wedge l_n$，　$l_j = x_j$ 或 $\neg x_j$
- 写出2个变量的全部函数的析取范式（一共有$2^{2^n}=2^4= 16$个）
  - 【注意：另一套等价的与、或、非记号；计算系统思维常用】
  - $y = 0$ 【注】　　$y = \overline{x_1}\cdot\overline{x_2} + \overline{x_1}\cdot x_2 + x_1\cdot\overline{x_2} + x_1\cdot x_2 = 1$
  - $y = x_1\cdot\overline{x_2} + x_1\cdot x_2 = x_1$　　$y = \overline{x_1}\cdot x_2 + x_1\cdot\overline{x_2} + x_1\cdot x_2 = x_1 + x_2$　　……

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**0**

x_1	x_2	y
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Boolean Algebra

- 另一种枚举布尔函数的方法：
  - 从布尔代数(L, +, •, ¬, 0, 1)获得**布尔表达式**
  - L is the set of all Boolean expressions.
    - $0, 1, x_1, \ldots, x_n \in$ L; 初始假定
    - If $x, y \in$ L, then $\neg x, x \bullet y, x + y \in$ L. 递归直至达到闭包
      - 且满足如下公理（用下列公理消除等价表达式）

1.	Associativity:	$(x \bullet y) \bullet z = x \bullet (y \bullet z)$, $(x + y) + z = x + (y + z)$	=2•3•5 =2+3+5	**结合律**
2.	Commutativity:	$x \bullet y = y \bullet x$, $x + y = y + x$	=2•3 =2+3	**交换律**
3.	Distributivity:	$(x + y) \bullet z = (x \bullet z) + (y \bullet z)$, $(x \bullet y) + z = (x + z) \bullet (y + z)$	(2+3)•5 =2•5+3•5 (2•3)+5 ≠(2+5)•(3+5)	**分配率**
4.	Identity: Annihilator:	$x + 0 = x$, $x \bullet 1 = x$ $x \bullet 0 = 0$, $x + 1 = 1$	2+0=2, 2•1=2 2•0=0, 2+1≠1	**有界律**
5.	Idempotence:	$x \bullet x = x$, $x + x = x$	2•2 ≠ 2, 2+2 ≠ 2	**幂等律**
6.	Absorption:	$(x \bullet y) + x = x$, $(x + y) \bullet x = x$	(2•3)+2≠2, (2+3)•2≠2	**吸收律**
7.	Complementation:	$x + \neg x = 1$, $x \bullet \neg x = 0$		**互补律**

中学代数，设
x, y, z = 2, 3, 5

(排中律)
非真既假

2个变量的布尔表达式

- 第一轮。常数和变量及其非，共6个表达式：

$$0, 1, x_1, x_2, \overline{x_1}, \overline{x_2}$$

- 第二轮。应用与或非到第一轮结果，使用公理整理，共8个新表达式：

$$\overline{x_1} + \overline{x_2}, \quad \overline{x_1} + x_2, \quad x_1 + \overline{x_2}, \quad x_1 + x_2$$
$$\overline{x_1} \cdot \overline{x_2}, \quad \overline{x_1} \cdot x_2, \quad x_1 \cdot \overline{x_2}, \quad x_1 \cdot x_2$$

- 第三轮。应用与或非到第二轮表达式，使用公理整理，共2个新表达式：

$$\overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2, \quad \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}$$

- 第四轮。应用与或非到第三轮表达式，使用公理整理，共0个新表达式。达到闭包。

2个变量的布尔表达式所对应的真值表

- 第一轮。常数和变量及其非，共6个表达式：$0, 1, x_1, x_2, \overline{x_1}, \overline{x_2}$

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**0**

x_1	x_2	y
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**0**
1	1	**1**

x_1	x_2	y
0	0	**1**
0	1	**1**
1	0	**0**
1	1	**0**

x_1	x_2	y
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**0**

- 第二轮。8个新表达式：$\overline{x_1} + \overline{x_2}$，$\overline{x_1} + x_2$，$x_1 + \overline{x_2}$，$x_1 + x_2$　$\overline{x_1} \cdot \overline{x_2}$，$\overline{x_1} \cdot x_2$，$x_1 \cdot \overline{x_2}$，$x_1 \cdot x_2$

x_1	x_2	y
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

x_1	x_2	y
0	0	**1**
0	1	**1**
1	0	**0**
1	1	**1**

x_1	x_2	y
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**0**
1	1	**0**

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**0**

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

- 第三轮。共2个新表达式：$\overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2$，$\overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}$

x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

x_1	x_2	y
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

思考题

- n个输入变量1个输出变量的布尔函数有多少个？
  - n=0：2个（0，1）
  - n=1：4个（1，0，x，$\neg x$）
  - n=2：**？**
  - …
  - n=n：$\mathbf{2^{2^n}}$
- 布尔函数等价于布尔表达式
  - 每一个真值表对应的布尔函数，存在等价的布尔表达式
  - 反之亦然
- 系统思维：组合电路能实现任意布尔函数吗？
  - 组合电路：由与、或、非门电路组合而成的逻辑电路

Kleene函数真值表（是一个三值逻辑）

x	y	$x \wedge y$	$x \vee y$	$\neg x$
0	0	0	0	1
0	I	0	I	1
0	1	0	1	1
I	0	0	I	I
I	I	I	I	I
I	1	I	1	I
1	0	0	1	0
1	I	I	1	0
1	1	1	1	0

Kleene函数是数学函数

$f:\{0, \mathrm{I}, 1\}^n \rightarrow \{0, \mathrm{I}, 1\}$

值	含义
0	False
I	Indeterminate
1	True

排中律不再成立

$x \vee \neg x \neq 1$

I	1/2
$x \wedge y$	min(x,y)
$x \vee y$	max(x,y)
$\neg x$	1-x

Kleene Algebra（替换布尔代数的排中律）

- 从克莱因代数 (L, +, •, ¬, 0, 1)获得克莱因表达式
  - L is the set of all Kleene expressions.
    - $0,1, x_1, \ldots, x_n \in$ L; 初始假定
    - If $x, y \in$ L, then $\neg x, x \bullet y, x + y \in$ L. 递归直至达到闭包
      - 且满足如下公理（用下列公理消除等价表达式）

1.	Associativity:	$(x \bullet y) \bullet z = x \bullet (y \bullet z)$, $(x + y) + z = x + (y + z)$	**结合律**
2.	Commutativity:	$x \bullet y = y \bullet x$, $x + y = y + x$	**交换律**
3.	Distributivity:	$(x + y) \bullet z = (x \bullet z) + (y \bullet z)$, $(x \bullet y) + z = (x + z) \bullet (y + z)$	**分配率**
4.	Identity: Annihilator:	$x + 0 = x$, $x \bullet 1 = x$ $x \bullet 0 = 0$, $x + 1 = 1$	**有界律**
5.	Idempotence:	$x \bullet x = x$, $x + x = x$	**幂等律**
6.	Absorption:	$(x \bullet y) + x = x$, $(x + y) \bullet x = x$	**吸收律**
7.	de Morgan:	$\neg(x+y) = \neg x \bullet \neg y$, $\neg(x \bullet y) = \neg x + \neg y$	**摩根律**
8.	Double Negation:	$\neg\neg x = x$	**对合律**
9.	Product:	$p \bullet x_i + p \bullet \neg x_i = p + p \bullet x_i + p \bullet \neg x_i$ where p is a product containing $\neg x_j \bullet x_i$	

在布尔代数中，
$p \bullet x_i + p \bullet \neg x_i = p$

克莱因表达式 ≠ Kleene函数

- An *n*-variable Kleene function (≠ Kleene expression)
  - $f:\{0,\mathrm{I},1\}^n \to \{0,\mathrm{I},1\}$
- 两个Kleene表达式相同：
  - 有相同的真值表
  - 能从Kleene公理集推出
- 一个输入变量*x*的Kleene表达式
  - 一共六个，而不是3^{3^1}=27个

x	$\overline{x}$	$x \bullet \overline{x}$	$x + \overline{x}$
0	1	0	1
I	I	I	I
1	0	0	1

- 第一轮。常数和变量及其非，共4个表达式：$0, 1, x, \overline{x}$
- 第二轮。应用与或非到第一轮结果，使用公理整理，共2个新表达式：(布尔逻辑中，这两个表达式不是新的)

$$x \cdot \overline{x}, \qquad x + \overline{x}$$

- 第三轮。应用与或非到第二轮表达式，使用公理整理，共0个新表达式。达到闭包。

x	y
0	0
I	0
1	0

$y = 0$

x	y
0	1
I	1
1	1

$y = 1$

x	y
0	0
I	I
1	1

$y = x$

x	y
0	1
I	I
1	0

$y = \overline{x}$

x	y
0	0
I	I
1	0

$y = x \cdot \overline{x}$

x	y
0	1
I	I
1	1

$y = x + \overline{x}$

x	y
0	0
I	I
1	I

$y = \mathrm{I}$

计算机科学很幸运地选择了布尔逻辑

- n个变量的布尔函数（布尔表达式）有多少个？
  - n=0：2个（0和1）
  - n=1： 4个（1，0，x和¬ x ）
  - ？
  - n=n： 2^{2^n}。所有布尔函数都是布尔表达式。

不存在真值表
无对应表达式

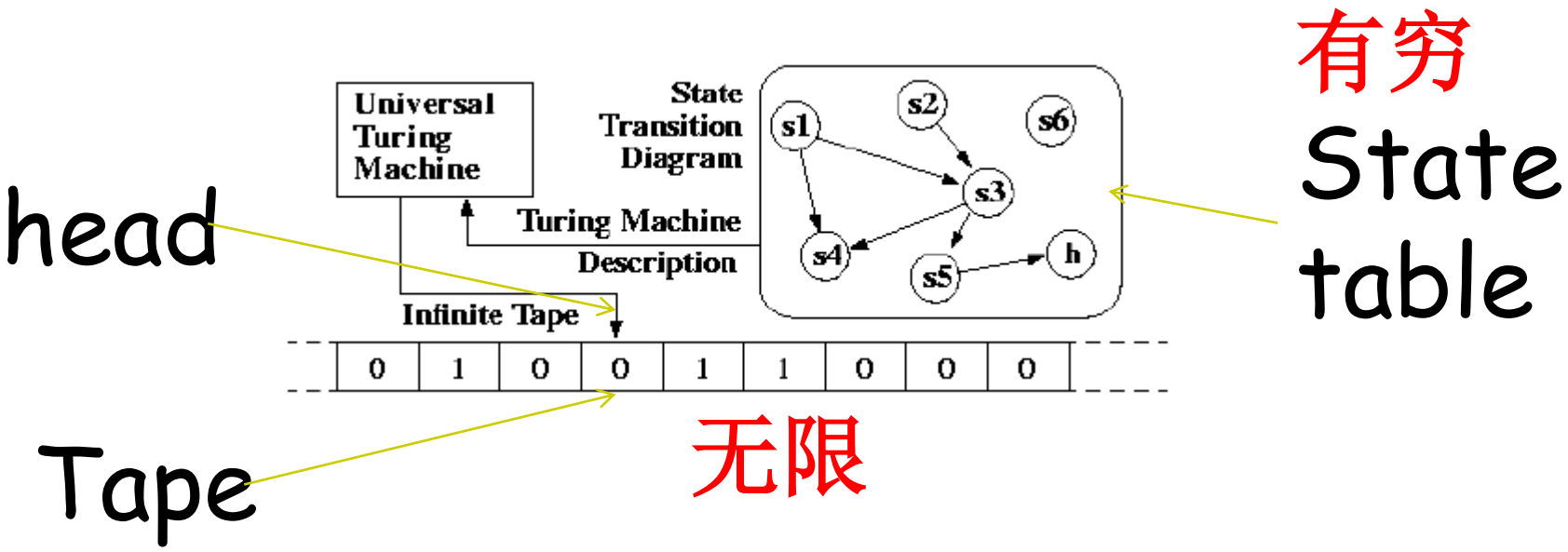
- n个变量的克莱因表达式有多少个？
  - 不是3^{3^n}；尚未找到闭合公式，但知道 $< 2^{3^n}$

存在真值表
无对应表达式

n	布尔函数个数 2^{2^n}	布尔表达式个数 2^{2^n}	克莱因表达式个数	克莱因函数个数 3^{3^n}
1	$2^{2^1}=4$	$2^{2^1}=4$	6	$3^{3^1}=27$
2	$2^{2^2}=16$	$2^{2^2}=16$	84	$3^{3^2}=3^9$
3	$2^{2^3}=256$	$2^{2^3}=256$	43918	$3^{3^3}=3^{27}$
4	$2^{2^4}=65536$	$2^{2^4}=65536$	160297985276	$3^{3^4}=3^{81}$

2.2 图灵机(Turing Machine)

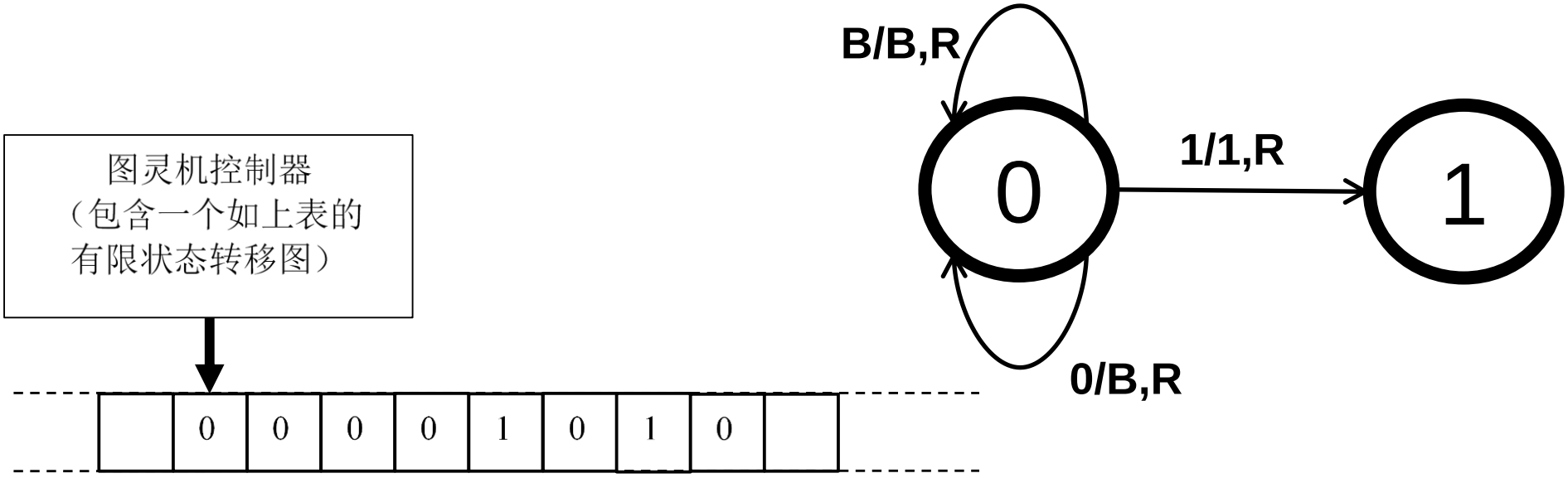
教科书234-235页还有一个例子，请同学们操作一遍



...an unlimited memory capacity obtained in the form of an infinite tape marked out into squares, on each of which a symbol could be printed. At any moment there is one symbol in the machine; it is called the scanned symbol. The machine can alter the scanned symbol and its behavior is in part determined by that symbol, but the symbols on the tape elsewhere do not affect the behavior of the machine. However, the tape can be moved back and forth through the machine, this being one of the elementary operations of the machine. Any symbol on the tape may therefore eventually have an innings. (by Turing 1948)

实例：“擦0停1”图灵机

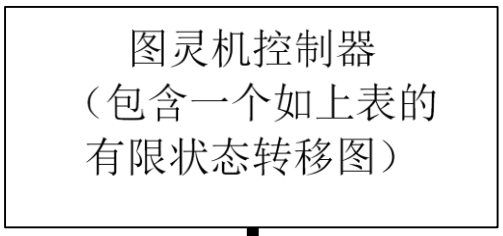
当前状态	读到的符号	写上的符号	读写头移动	下一状态
0	空白	空白	右	0
0	0	空白	右	0
0	1	1	右	1（终止）

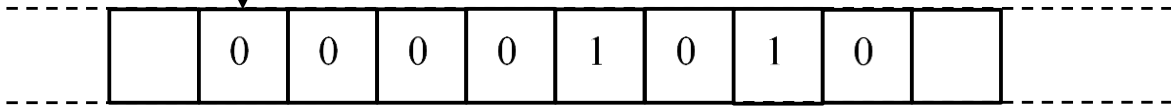


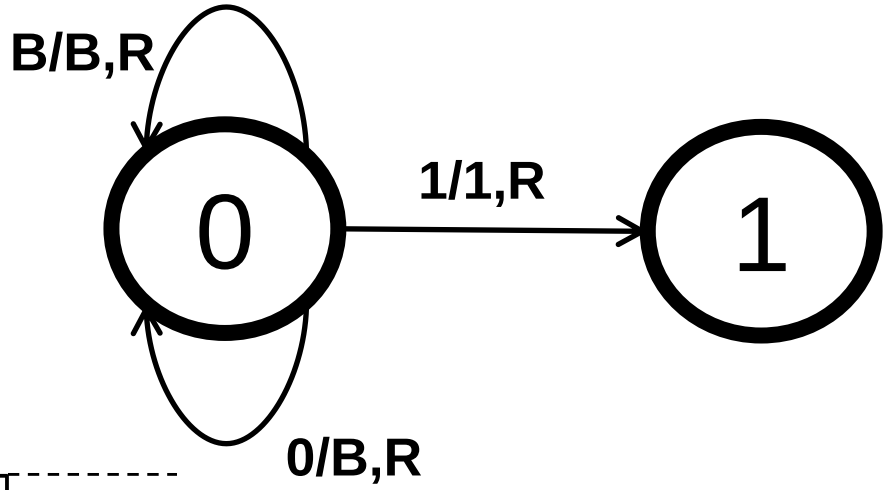
实例：“擦0停1”图灵机

当前状态	读到的符号	写上的符号	读写头移动	下一状态
0	空白	空白	右	0
0	0	空白	右	0
0	1	1	右	1（终止）

第四步后，状态转移表到哪里？纸带状态是什么？







π是可计算数”吗？

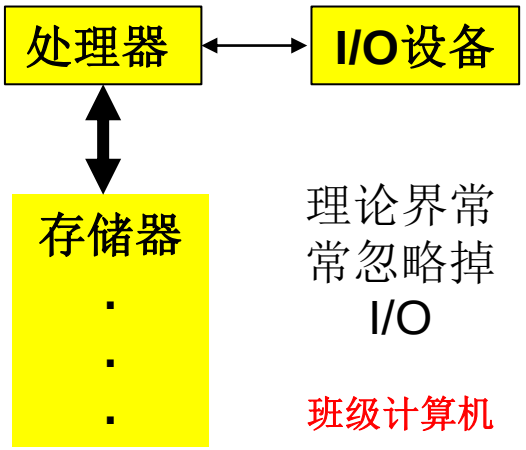
- Computable numbers are real numbers whose **decimals are calculable by finite machines**

- 有限**Go**程序计算了π的前**800**个十进制数位

**3141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318505820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318

2.3 图灵机的通用性和局限性

- **计算机能够求解任意可计算问题（称为功能）**
  - 有限性：真实计算机只有有限精度和有限存储
  - 科学计算、企业计算、消费者计算
- **丘奇-图灵论题（Church-Turing Hypothesis）**
  - 人用纸和笔所能做的计算 与 图灵机能自动执行的计算 等价
  - （无限存储器）冯诺依曼模型计算机与图灵机等价
- **存在不可计算问题**
  - 例如：
    - 停机问题



2.4 Godel不完备性定理

- 定理一：任意一个包含一阶谓词逻辑与初等数论的形式系统，都不可能同时拥有完备性和一致性。即存在一个真命题，它在这个系统中不能被证明。
- 定理二：任意一个包含初等数论的系统S，当S无矛盾时，它的无矛盾性不可能在S内证明。

真和可以被证明是两件事情!!

Kurt Godel

1906-1978

Godel不完备性定理的实例：戈德斯坦定理

哥德尔不完备定理的一个实例

- 哥德尔不完备定理：在任何包含初等算术的数学系统中，存在真但不可证的命题。
  - 初等算术=皮亚诺算术
  - 哪个命题？
- Goodstein Theorem, 1944
  - Every Goodstein sequence approaches zero.
- Kirby & Paris, 1982
  - 戈德斯坦定理不能在皮亚诺算术中证明.
- Caicedo, 2007
  - Kirby & Paris结论的新的构造性证明.

皮亚诺算术五公理
(引自大不列颠百科全书)

1. Zero is a natural number.
   零是自然数
2. Every natural number has a successor in the natural numbers.
   任意自然数有一后继自然数
3. Zero is not the successor of any natural number.
   零不是任何自然数的后继
4. If the successor of two natural numbers is the same, then the two original numbers are the same.
   后继相同的两个自然数相等
5. If a set contains zero and the successor of every number is in the set, then the set contains the natural numbers. 归纳公理

Andrés Eduardo Caicedo, Goodstein's function. Revista Colombiana de Matemáticas 41 (2) (2007), 381–391.

Goodstein sequence for n=4

郭泓锐的结果（计算机，2017）

- 提出了新算法，指数性降低计算复杂度
  - 1400行Go程序，在单节点服务器上算了13小时
- 计算结果如下，注意$G(4) = 3\cdot 2^{402653211}-2$，最大数所在下标$k= 3\cdot 2^{402653209}$

$(4)_0 =$ 4

$(4)_1 =$ 26

$(4)_2 =$ 41

$(4)_3 =$ 60

$(4)_4 =$ 83

$$\text{最大数}(4)_K = 3.447540401546 31\ldots \times 10^{121210695}$$

.....

$(4)_{k-2} =$ 这五个最大的数有下面两个性质：字长是121210695十进制位，最高位1000个十进制数字是

$(4)_{k-1} =$ 3447540401546310082868194979805754978478874937901486829482582471181371747989862435944126537723138836357706343870670981471373770123119725827117104237084886899557319167763456466003661752256536556670766073663822155027872496625257503330885348667848633411493190314615269655969736866492160948225290436847365886709876147562092042005868664973311829175856332138120219517198418201812335339301056271348711228774295067529019486998025111083607801145279169927216822914248107894561933408544103589430851495052430471521491596915665031176899651610957212217360780656154707158846933785793375188967822229622822797778043761152773386718099235161662127038925419805394792980981939486485522909222878857883887560348367316381270673280675347382769219015375432616565108108081814310923711203313305968399971676957781217795694754003625391588939038853733479876344772723623575017620929937195505529441455741049571737770925493092772866804132388313424521454495162309274452552557713106522747593529930053066060085629731708801300896843377521210695

$(4)_{k+1} =$

$(4)_{k+2} =$

.....

$(4)_{G(4)-1} =$ 4

$(4)_{G(4)-1} =$ 3

$(4)_{G(4)-1} =$ 2

$(4)_{G(4)-1} =$ 1

$(4)_{G(4)} =$ 0

3. 算法思维

- **高德纳的算法定义：**一个算法是一组有穷的规则，给出求解特定类型问题的运算序列，并具备下列五个特征：

（1）有穷性：一个算法在有限步骤之后必然要终止。

（2）确定性：一个算法的每个步骤都必须精确地（严格地和无歧义地）定义。

（3）输入：一个算法有零个或多个输入。

（4）输出：一个算法有一个或多个输出。

（5）能行性：一个算法的所有运算必须是充分基本的，原则上人们用笔和纸可以在有限时间内精确地完成它们。

算法思维成功例子

- 分治思想
  - **各种排序**
  - 单因素优选法
  - 大整数乘法
  - 矩阵乘法
- 其他方法：贪心、穷举、动态规划、二分法
  - **快速排序**

算法的复杂度分析

- 求解斐波那契数列的递归算法GO代码耗时多少？

```
func fibonacci(n int) int {
        if n == 0 || n == 1 {
                return n
        }
        return fibonacci(n-1) + fibonacci(n-2)
}
```

- 复杂度分析（常用递推公式）
  - T(n) = T(n-1) + T(n-2)
    - 2*T(n-2) < T(n) < 2*T(n-1), 当n > 2时
  - T(n) < 2*T(n-1) < 4*T(n-2) < 8*T(n-3) … < 2^n
  - T(n) > 2*T(n-2) > 4*T(n-4) > 8*T(n-6) … > $2^{n/2}$
  - **T(n) = O(2^n)， T(n) = Ω($2^{n/2}$)**　　**指数复杂度！**
- 动态规划算法：复杂度O(n)

小o，大O， Ω，Θ记号

共同假设：当n足够大

- $f(n) = o(g(n))$ $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
  - $n^{1.58} = \mathbf{o}(n^2)$， $n^{1.58} \neq \mathbf{o}(n^{1.58})$ ， $n^2 \neq \mathbf{o}(n^{1.58})$
- $f(n) = O(g(n))$ ∃常数$c > 0, f(n) \leq cg(n)$
  - $n^{1.58} = \mathbf{O}(n^2)$， $n^{1.58} = \mathbf{O}(n^{1.58})$， $n^2 \neq \mathbf{O}(n^{1.58})$
- $f(n) = \Omega(g(n))$ ∃常数$c > 0, f(n) \geq cg(n)$
  - $n^{1.58} \neq \mathbf{\Omega}(n^2)$， $n^{1.58} = \mathbf{\Omega}(n^{1.58})$， $n^2 = \mathbf{\Omega}(n^{1.58})$
- $f(n) = \Theta(g(n))$ $f(n) = O(g(n))$ 并且 $f(n) = \Omega(g(n))$
  - $n^{1.58} \neq \mathbf{\Theta}(n^2)$， $n^{1.58} = \mathbf{\Theta}(n^{1.58})$， $n^2 \neq \mathbf{\Theta}(n^{1.58})$

小o，大O，Ω，Θ记号

共同假设：当n足够大。g是f的严格上阶、上阶、下阶、等阶

- $f(n) = o(g(n))$ **g是f的严格上界（阶）** $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
  - $n^{1.58} = \mathbf{o}(n^2)$， $n^{1.58}$ ？ $\mathbf{o}(n^{1.58})$， $n^2 \neq \mathbf{o}(n^{1.58})$
- $f(n) = O(g(n))$ 上界 ∃常数$c > 0, f(n) \leq cg(n)$
  - $n^{1.58}$ ？ $\mathbf{O}(n^2)$， $n^{1.58} = \mathbf{O}(n^{1.58})$， $n^2 \neq \mathbf{O}(n^{1.58})$
- $f(n) = \Omega(g(n))$ 下界 ∃常数$c > 0, f(n) \geq cg(n)$
  - $n^{1.58}$ ？ $\mathbf{\Omega}(n^2)$， $n^{1.58} = \mathbf{\Omega}(n^{1.58})$， $n^2 = \mathbf{\Omega}(n^{1.58})$
- $f(n) = \Theta(g(n))$ 等阶 $f(n) = O(g(n))$ 并且 $f(n) = \Omega(g(n))$
  - $n^{1.58}$ ？ $\mathbf{\Theta}(n^2)$， $n^{1.58} = \mathbf{\Theta}(n^{1.58})$， $n^2 \neq \mathbf{\Theta}(n^{1.58})$

NP vs P

- 输入2*n*个数，判断是否可以把这些数等分成两组（每组n个数），使得两组的和相同。
- 是否是NP的？
  - 是！
  - 为什么？
    - 给定结果，即满足题意的分组方式
    - 验证算法：验证每组都是n个数，且两组数的和相等
    - **验证算法是多项式的**（事实上验证算法的复杂度是O(n)）！
- 是否是P的？
  - 目前不知道。如果找到多项式时间算法，则NP=P。

NP vs P

- 输入2*n*个数，判断是否可以把这些数等分成两组（每组n个数），使得两组的和不相同。
- 是否是NP的？
  - 是！方法类似之前。
- 是否是P的？
  - 是！
  - 只要这2n个数不全相同，则答案为是。为什么？
  - O(n)

5. 计算系统思维

- 通过抽象，将**模块**组合成为系统，**无缝**执行计算过程
  - **抽象化**：一个通用抽象代表多个具体需求
  - **模块化**：系统由多个模块组合而成
    - 计算机 = 硬件 + 系统软件 + 应用软件
    - 全系统一致性；信息隐藏原理，接口概念
  - **无缝衔接**
    - 避免缝隙：
      - 扬雄周期原理、波斯特尔鲁棒性原理、冯诺依曼穷举原理
    - 重视瓶颈：阿姆达尔定律
      - 系统性能改进受限于系统瓶颈（针对某任务）
        - 加速比 = $1 / ((1-f)/p + f) \rightarrow 1/f$ 当 $p \rightarrow \infty$

本课程学到的一些基本抽象

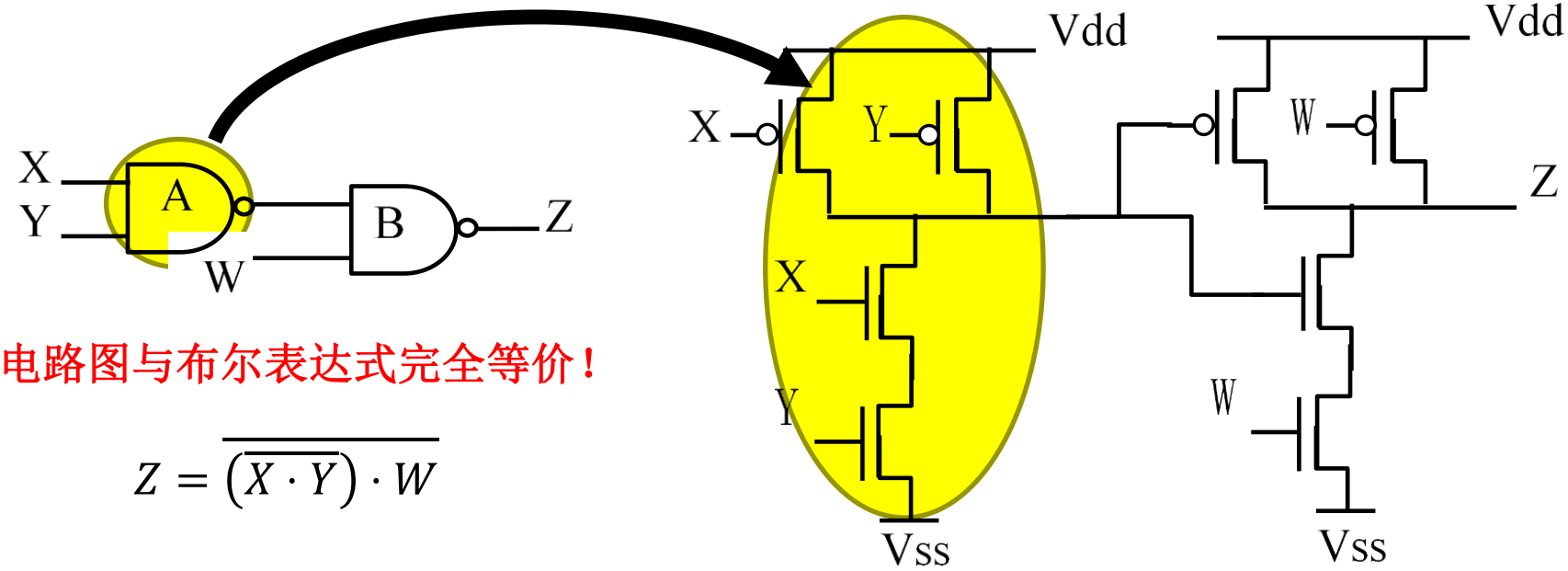
- 是什么、通过实例解释、能够举一反三

Data Type **数据类型**	bit (1 bit), hexadecimal number (4 bits), byte (8 bits), uint8 (8-bit unsigned integer), integer (64 bits); array (n elements of the same type), slice (a descriptor pointing to an array); text file, BMP image file; hypertext and hyperlink 比特、字节、整数、数组、切片、文本文件、图像文件、超链接	
Software **软件**	Algorithm 算法	Smart method of information transformation, such as quicksort, hiding text in a BMP file, etc. 快排算法
	Program 程序	Code realizing algorithms in computer language, such as hide.go in the Text Hider project 信息隐藏程序 **hide.go**
	Process 进程	Program in execution, such as the "hide" process running in a Linux environment **> ./WebServer &** 系统显示 PID=79
	Instruction 指令	The smallest unit of software, directly executable by computer hardware 班级快排指令集
von Neumann Architecture: a computer model bridging software and hardware 冯诺依曼计算机模型		
Hardware **硬件**	Instruction Pipeline 指令流水线	The basic hardware mechanism to automatically execute any instruction "取指-译码-执行"三级流水线
	Sequential Circuit 时序电路	More precisely, only consider Synchronous Sequential Circuit comprised of combinational circuits and state circuits and driven by a clock signal; equivalent to the automata concept 串行加法器
	Combinational Circuit 组合电路	Also known as Boolean circuit, realizing a Boolean function 使用全加器的波纹进位加法器

信息隐藏原理实例

W	X	Y	Z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

- 图中三者表达同样的逻辑电路
- 但抽象不同，暴露的信息不同



门电路图与布尔表达式完全等价！

$$Z = \overline{(\overline{X \cdot Y}) \cdot W}$$

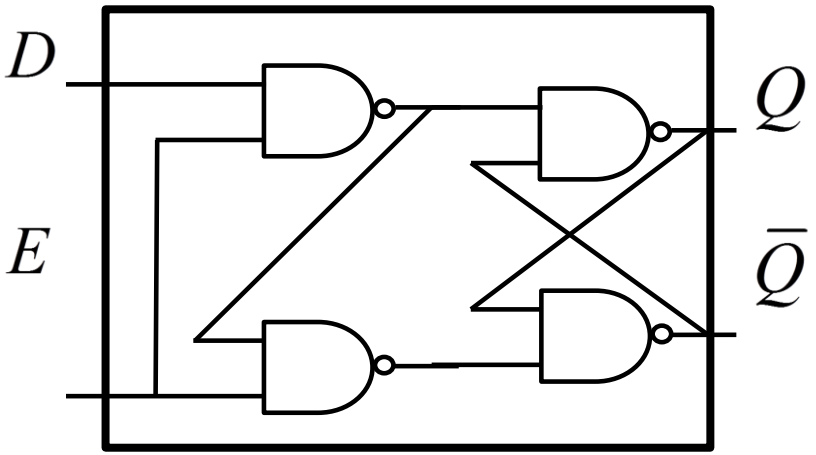
接口：逻辑值+逻辑操作　　电压值+晶体管操作

D-触发器

- Delay Flip-Flop
  - Enable往往用时钟信号CLK
  - 一拍时钟后，Q=D

E	D	Q	Q_{next}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

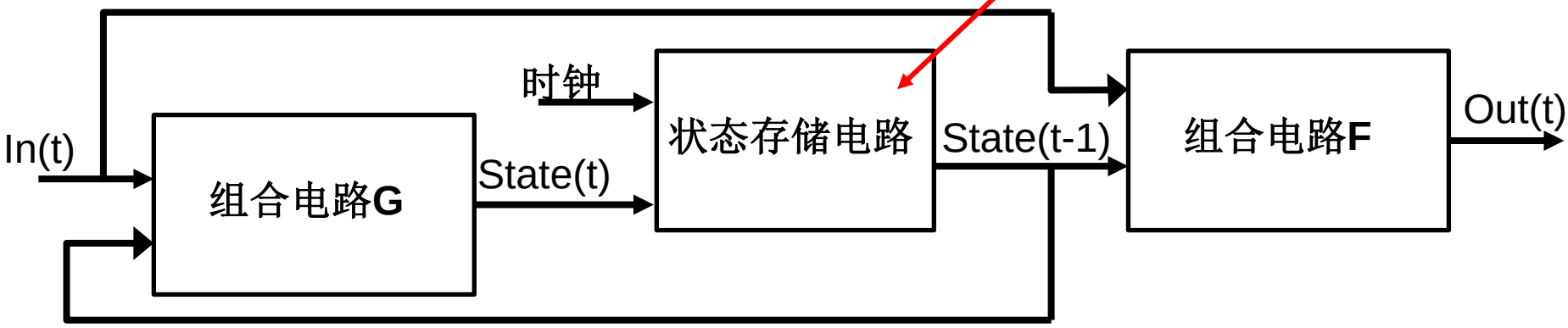
抽象化
隐藏内部细节



组合电路与状态电路产生自动机
即时钟同步的时序电路

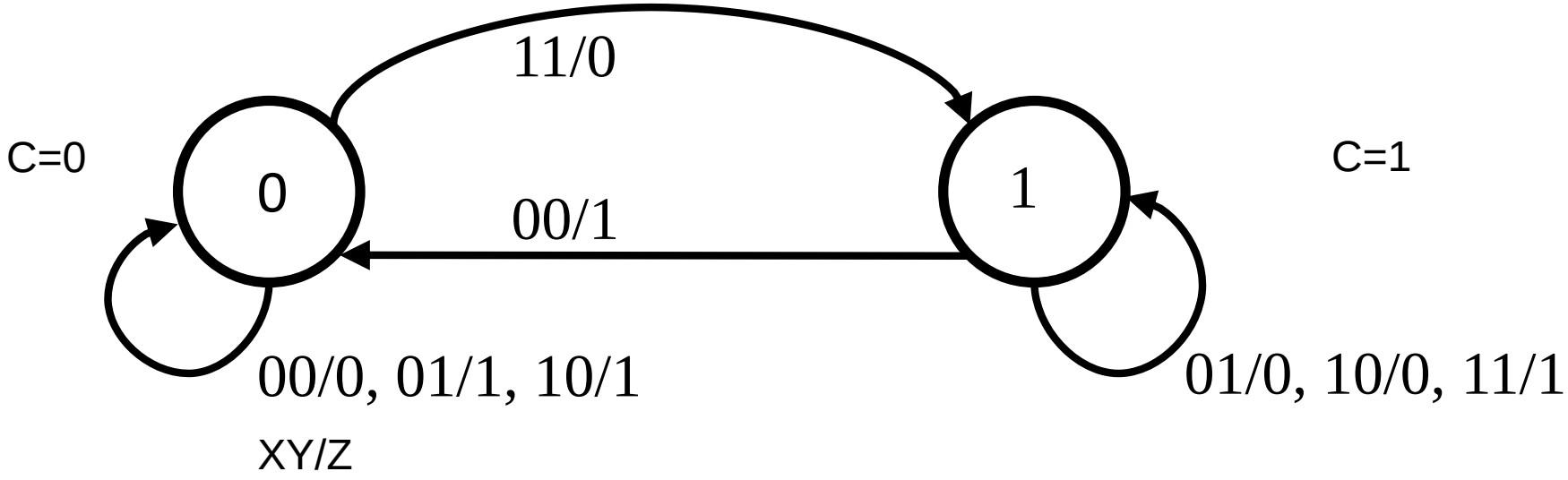
- 第t时刻的输出是t时刻输入与t-1时刻状态的函数
- 第t时刻的状态是t时刻输入与t-1时刻状态的函数
  - Out(t) = F(In(t), State(t-1))
  - State(t) = G(In(t), State(t-1))

往往用**D-触发器实现**



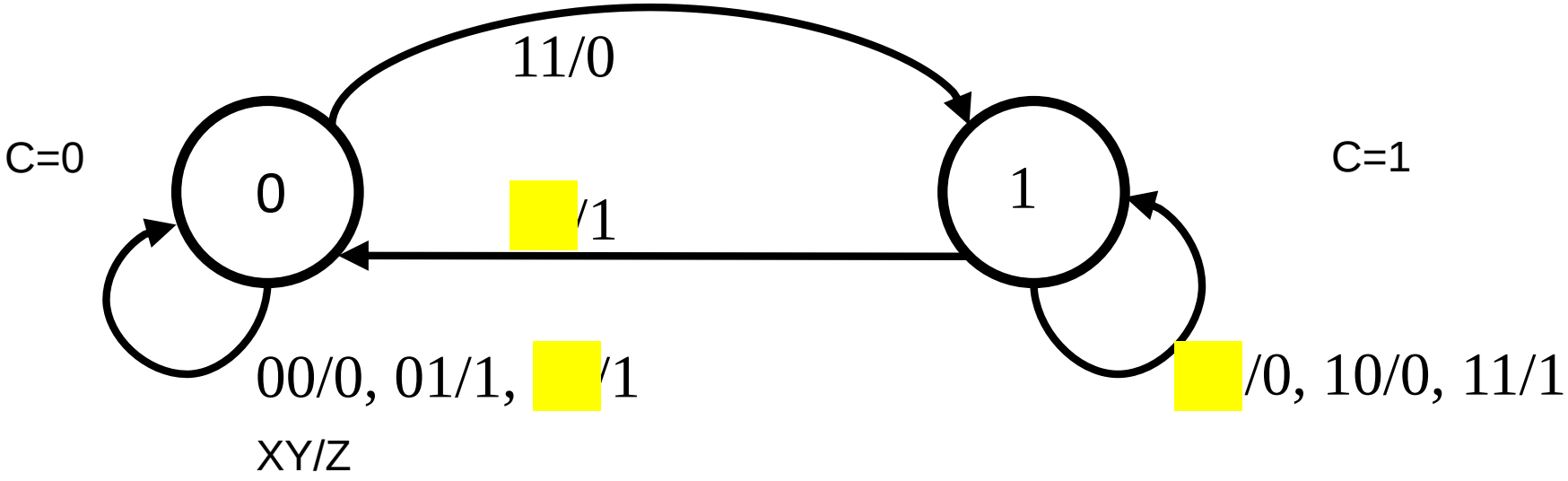
自动机实现4位串行加法过程

- 1011+1001 = 10100
  - t=0的初始状态：C=0。自动机处于左边状态。
  - t=1：X=1，Y=1；有向边11/0适用，自动机转移到右边状态，输出Z=0。
  - t=2：X=1，Y=0；有向边10/0适用，自动机保持在右边状态，输出Z=0。
  - t=3：X=0，Y=0；有向边00/1适用，自动机转移到左边状态，输出Z=1。
  - t=4：X=1，Y=1；有向边11/0适用，自动机转移到右边状态，输出Z=0。

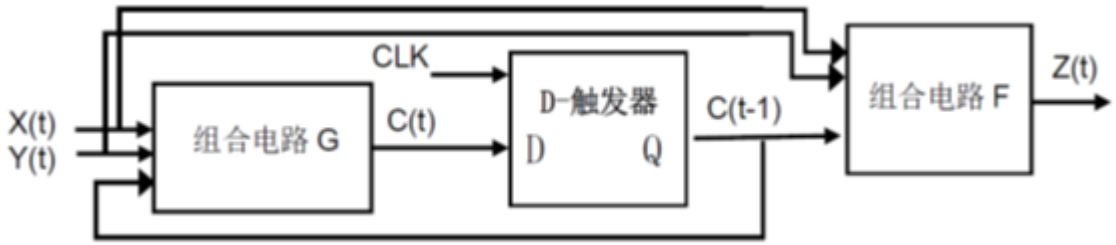


自动机实现4位串行加法过程

- 1011+1001 = 10100
  - t=0的初始状态：C=0。自动机处于左边状态。
  - t=1：X=1，Y=1；有向边11/0适用，自动机转移到右边状态，输出Z=0。
  - t=2：X=1，Y=0；有向边10/0适用，自动机保持在右边状态，输出Z=0。
  - t=3：X=0，Y=0；有向边00/1适用，自动机转移到左边状态，输出Z=1。
  - t=4：X=1，Y=1；有向边11/0适用，自动机转移到右边状态，输出Z=0。



C	X	Y	C_{next}	Z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$Z = X \oplus Y \oplus C$$
$$C_{next} = (X \cdot Y) + (X \oplus Y) \cdot C$$

1011+1001 = 10100的4位加法过程如下：

- t=0的初始状态：C=0。
- t=1：X=1，Y=1；

$Z = 1 \oplus 1 \oplus 0 = \mathbf{0}, C = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$

- t=2：X=1，Y=0；

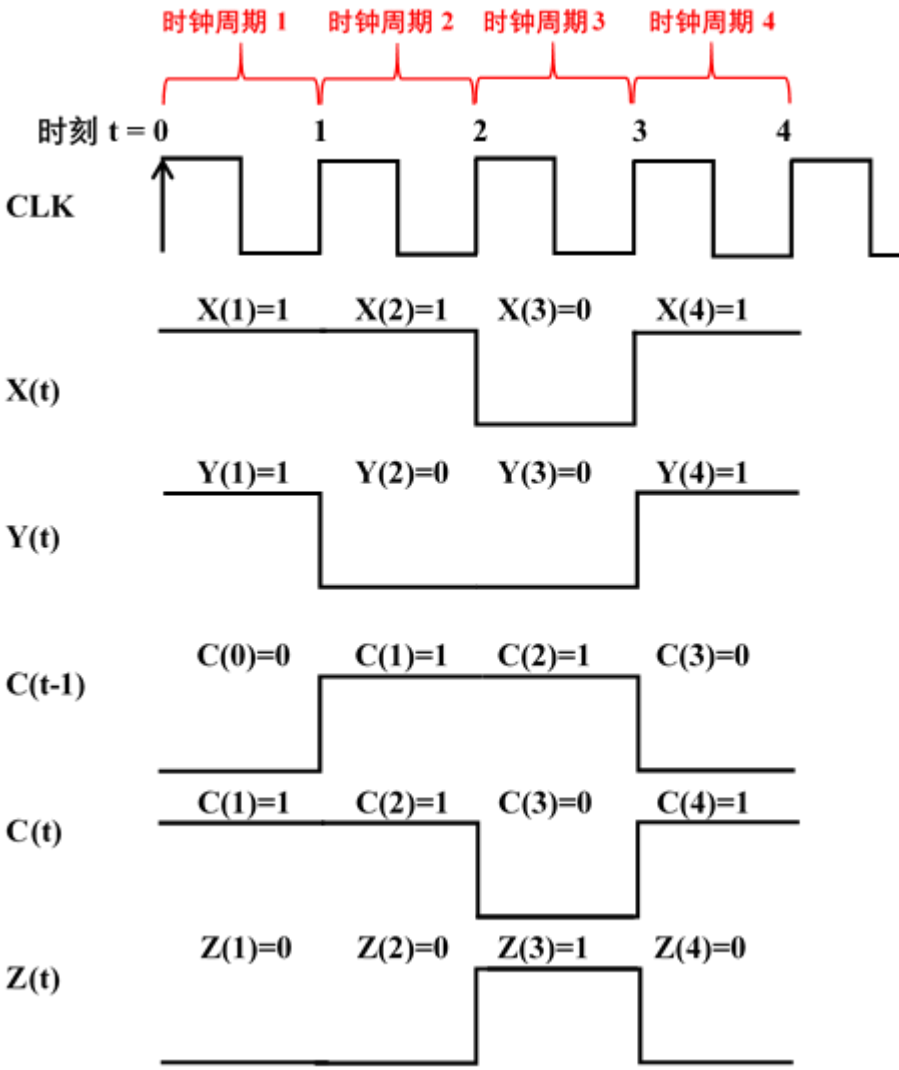
$Z = 1 \oplus 0 \oplus 1 = \mathbf{0}, C = (1 \cdot 0) + (1 \oplus 0) \cdot 1 = 1$

- t=3：X=0，Y=0；

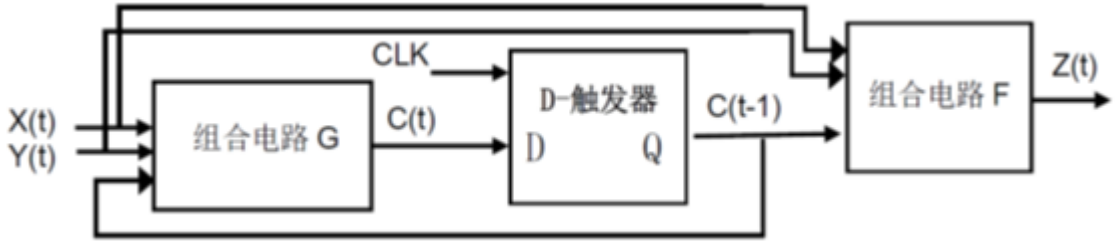
$Z = 0 \oplus 0 \oplus 1 = \mathbf{1}, C = (0 \cdot 0) + (0 \oplus 0) \cdot 1 = 0$

- t=4：X=1，Y=1；

$Z = 1 \oplus 1 \oplus 0 = \mathbf{0}, C = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = \mathbf{1}$



C	X	Y	C_{next}	Z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



减法如何做？

$$Z = X \oplus Y \oplus C$$
$$C_{next} = (X \cdot Y) + (X \oplus Y) \cdot C$$

1011+1001 = 10100的4位加法过程如下：

- t=0的初始状态：C=0。
- t=1：X=1，Y=1；

$Z = 1 \oplus 1 \oplus 0 = \mathbf{0}, C = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$

- t=2：X=1，Y=0；

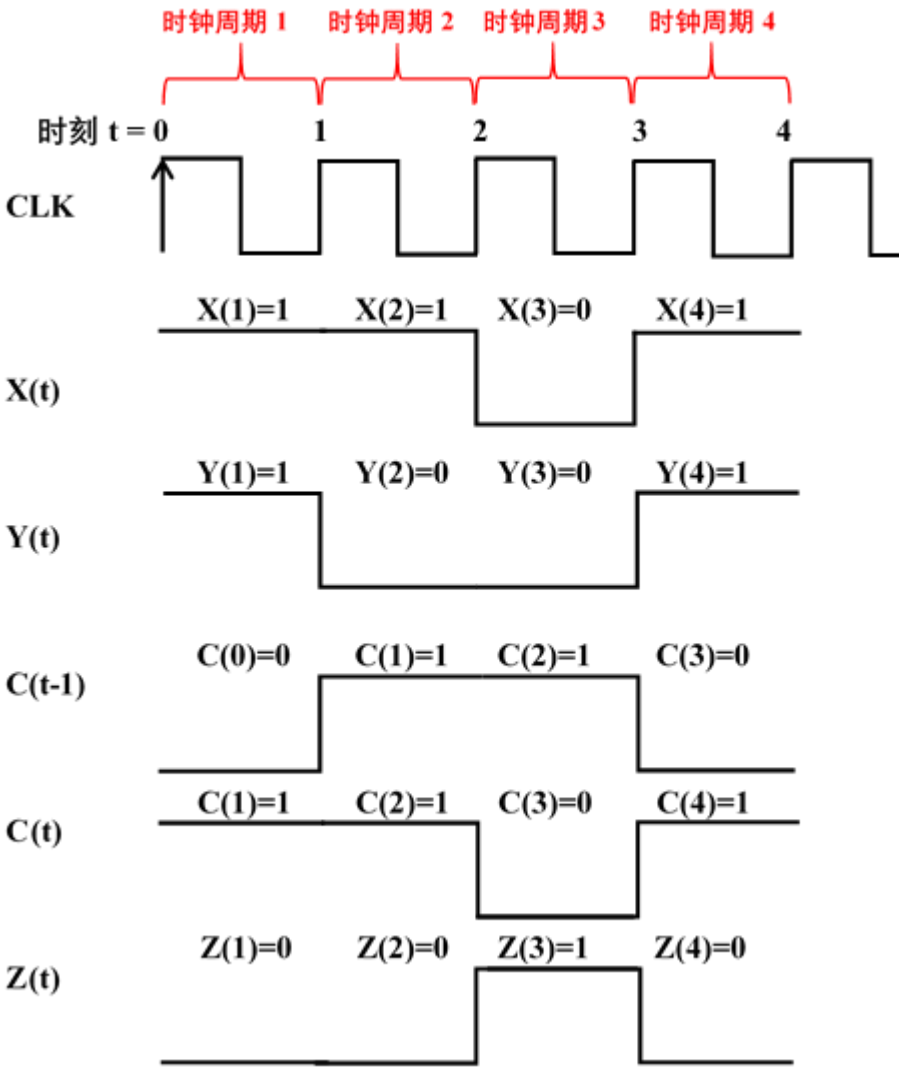
$Z = 1 \oplus 0 \oplus 1 = \mathbf{0}, C = (1 \cdot 0) + (1 \oplus 0) \cdot 1 = 1$

- t=3：X=0，Y=0；

$Z = 0 \oplus 0 \oplus 1 = \mathbf{1}, C = (0 \cdot 0) + (0 \oplus 0) \cdot 1 = 0$

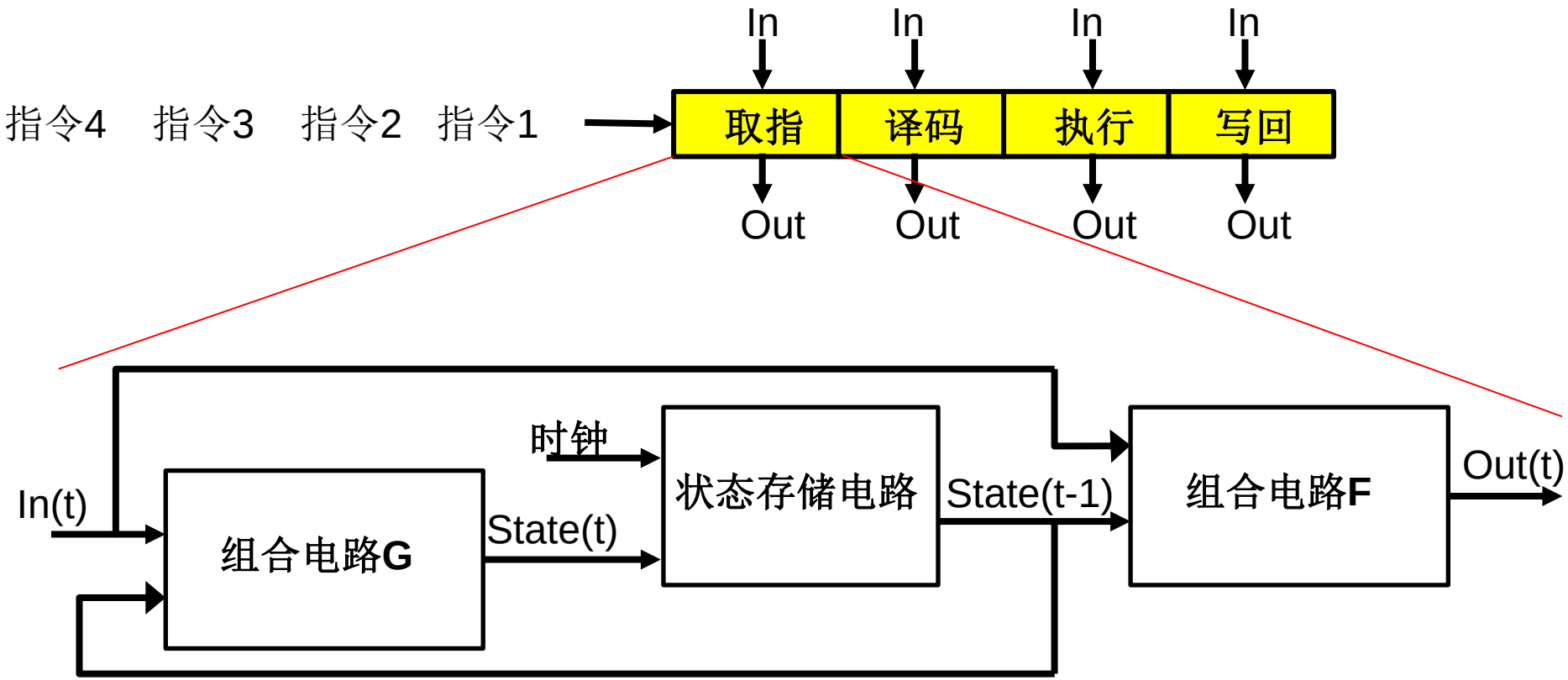
- t=4：X=1，Y=1；

$Z = 1 \oplus 1 \oplus 0 = \mathbf{0}, C = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = \mathbf{1}$



自动机、时序电路是系统基本概念
几乎无处不在

- 例如，任何程序、所有指令都是由指令流水线执行
  - 指令流水线是时钟同步的时序电路



软件

- 基础软件
  - 固件（firmware）：实现最基本的功能，通常“固化”在只读存储器芯片中
  - 系统软件：操作系统、编译器
  - 中间件：数据库软件、万维网服务器
- 应用软件
  - 办公软件、通信软件、行业软件、游戏软件、电子商务软件、上网软件等
- 二进制码：机器语言程序
- 源码：汇编语言或高级语言程序

自动执行与无缝衔接

- 自动执行高阶：无缝衔接
  - 扬雄周期原理
  - 波斯特尔鲁棒性原理
  - 冯诺依曼穷举原理
  - 阿姆达尔定律
- 自动执行难题尚未完全解决
  - 基本解决了单机自动执行难题
  - 在多台计算机上执行的计算过程如何自动执行？
  - 在人机物三元计算的万物互联网时代如何自动执行？
    - 网络思维（名字空间、拓扑、协议栈）
    - IEEE CS 2022报告：无缝智能

正常执行与异常处理

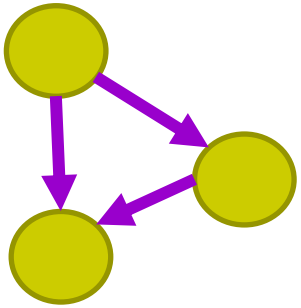
- 正常执行
  - 第一条指令在哪里？
  - 当前指令如何执行？
  - 下一条指令在哪里？
- 异常处理
  - 中断
    - 执行完毕当前指令，然后执行异常处理程序
  - 硬件出错
    - 立即执行一个事先设计好的异常处理程序
  - 保底异常
    - 为了做到穷举，保底异常（通常被称为machine check）覆盖其他异常没有覆盖的情况

重视瓶颈：阿姆达尔定律

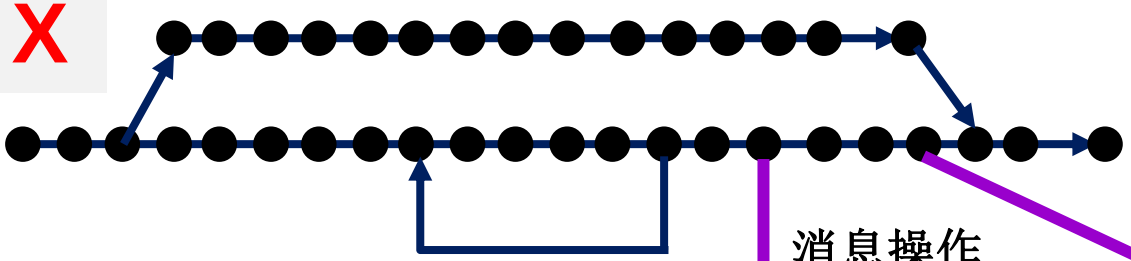
- 系统性能改进受限于系统瓶颈
  - 加速比 = 1 / ((1-f)/p + f) →1/f 当 p→∞
    - “假如一个系统可以分成两部分X和Y, X+Y=1, 0≤X≤1, 0≤Y≤1。Y能够改善（即缩小它的数值），X不能被改善（即X是瓶颈）。那么，系统最多能被改善到1/X。”
  - 一台电脑使用了500 MHz主频的处理器芯片，假设X＝Y＝0.5，即程序代码只有一半可以随处理器速度的增加而改善
    - 那么，我们将处理器的速度提高1000倍（提到500 GHz），整个电脑的速度提高多少？
      - 加速比只能变到1/0.5=2，即提高一倍

6. 网络思维

- 计算过程涉及（由多个**节点**连接而成的）**网络**
  - 网络成为计算过程的对象、执行系统
- 核心概念：连通性、消息传递、协议
  - **名字空间、拓扑、协议栈**



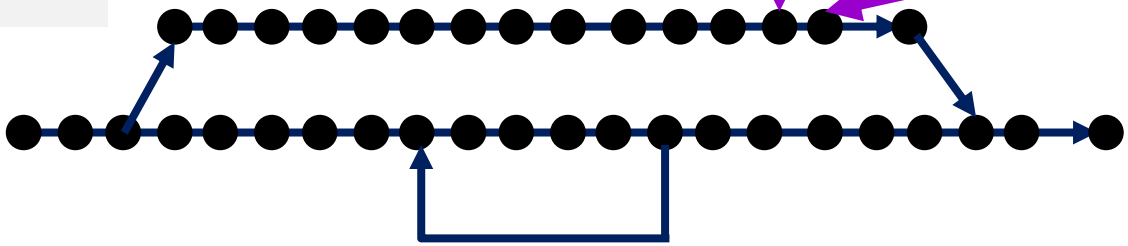
X



消息操作

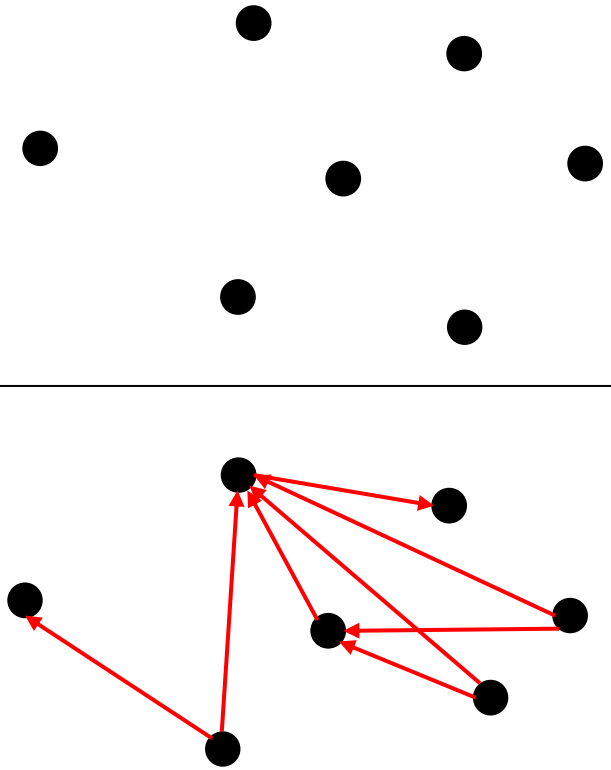
Z

Y



连通性与消息传递是松耦合关系

- 网络思维并不必涉及消息传递（或通信协议）
- 此时，重点是**连通性**（connectivity）
  - 即：有什么节点？节点之间如何连接？
  - 拓扑本身就有价值
- 搜索引擎实例
  - 第一代：无网络思维
    - 只关心节点的内容
  - 第二代：有网络思维
    - Page、 Kleinberg、李彦宏
    - 关心节点内容
    - **还关心网络拓扑（pagerank）**



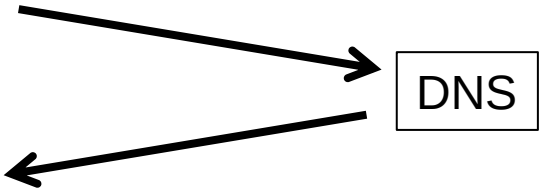
名字空间

- Name space；naming
- 主要用于指称网络中的节点

名字空间实例	节点的名字举例	名字空间解释
微信名字	中关村民	腾讯公司规定的任意“合法的”字符串
电子邮箱地址	zxu@ict.ac.cn	用户名@因特网域名
手机号码	189-8888-9999	通信公司规定的11位10进制数字串
本机文件路径（本地路径）	/我的文件/教材.pdf	本机操作系统规定的文件名
本机网卡地址（MAC地址）	00-1E-C9-43-24-42	全球统一规定的12位16进制数字串
网站域名	www.ict.ac.cn	互联网协议栈规定的域名
网站IP地址	159.226.97.84	IP协议规定的合法地址

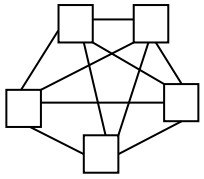
名字空间设计的两个问题

- 设计与理解名字空间的基本考虑
  - **唯一性**：zxu@ict.ac.cn　vs.　中关村民
    - 重用性：手机号码 vs. 万维网资源的URI
  - **自主性**：一个网卡可以插到另一台设备总线上吗？
    - 固定IP地址 vs. 动态生成IP地址
  - **友好性**：中关村民 vs. 以太网MAC地址
- 不同层次间的名字如何解析
  - 本地解析 vs. 全网解析（远程解析）
    - http://www.ict.ac.cn/本地路径...
    - http://159.226.97.84/本地路径...

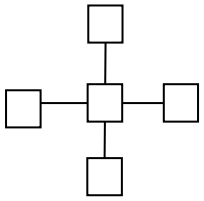


按动态性划分的三类网络拓扑

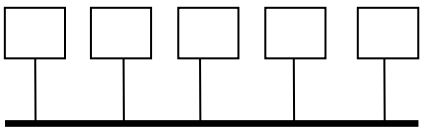
- 静态网络：节点完全确定、连接完全确定
- 动态网络：节点完全确定、连接部分确定
- 演化网络：节点部分确定、连接部分确定
- **你的微信朋友圈是什么网络？**



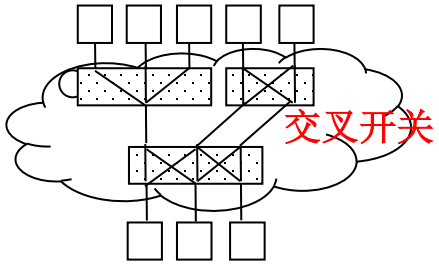
(a) 全连通图



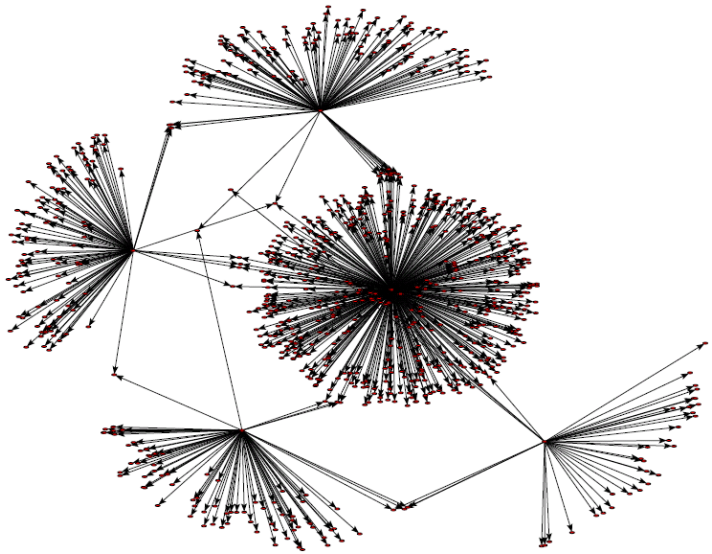
(b) 星型网络



(c) 总线



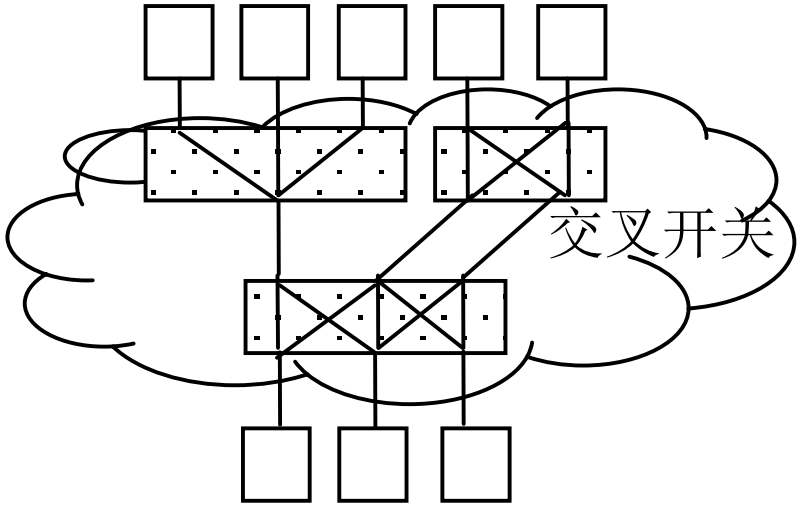
(d) 交换网络



(e) 演化网络

“到家了！”如何解析成底层的消息包？

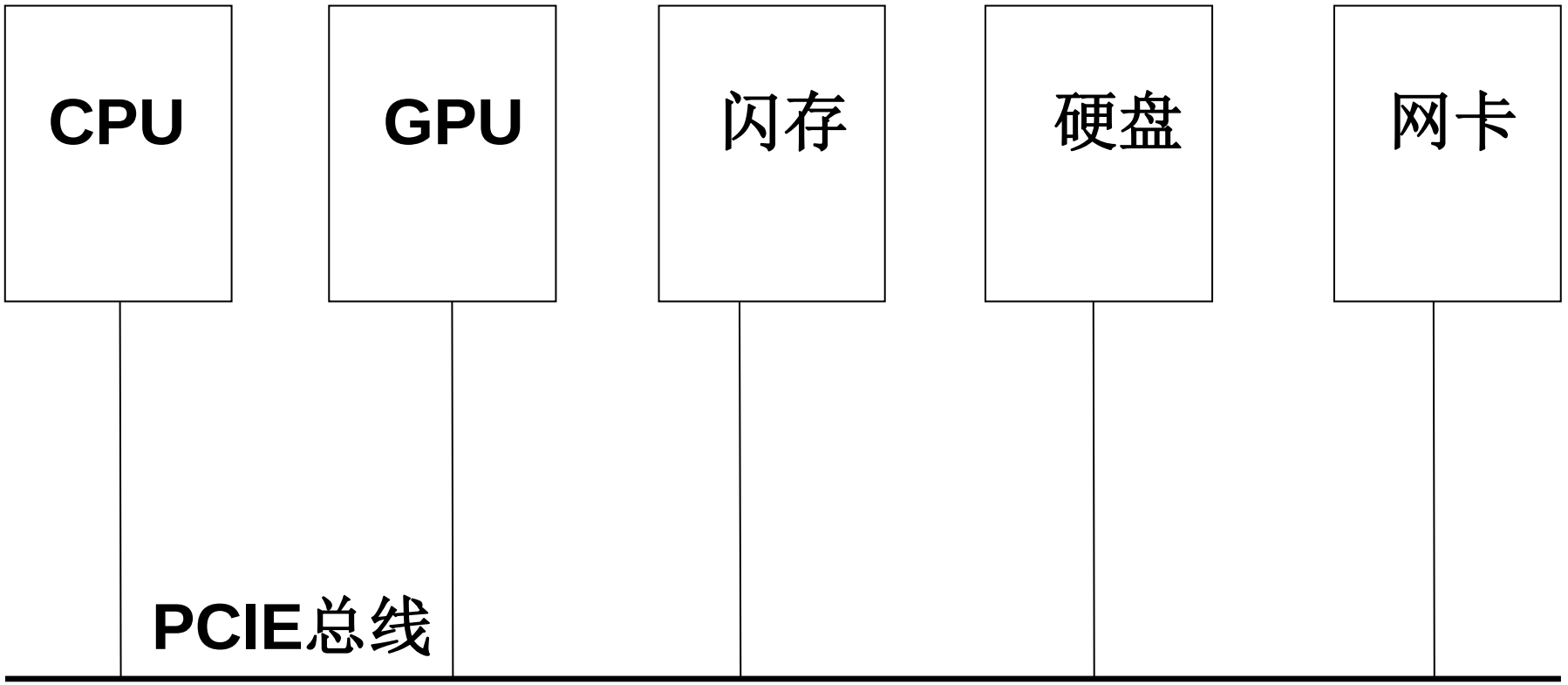
- 关键技术点
  - 分组交换
  - 包：首部+数据
  - 逐层解析并传输



微信协议：中关村民送消息“回家了！”给好友 Y

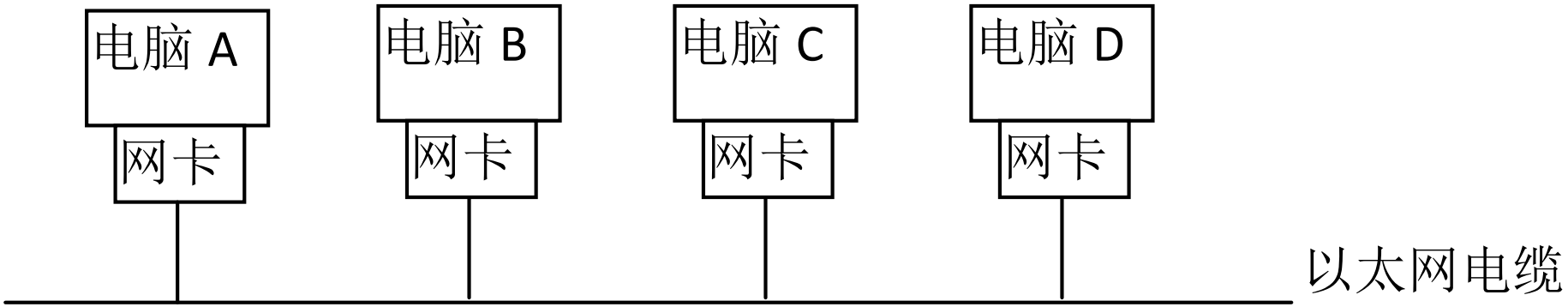
应用层		
应用层	HTTP	GET http://mp.weixin.qq.com/...回家了！...
传输层	TCP	TCP 首部 [80 端口] | GET :// /...回家...
网络层	IP	IP 首部 [140.206.160.199] | TCP 首部 | GET :// /...回家...
WiFi 层	802.11	WiFi 首部 | IP 首部 | TCP 首部 | GET :// /...回家...

在一个主板上的总线仲裁实例



局域网与以太网

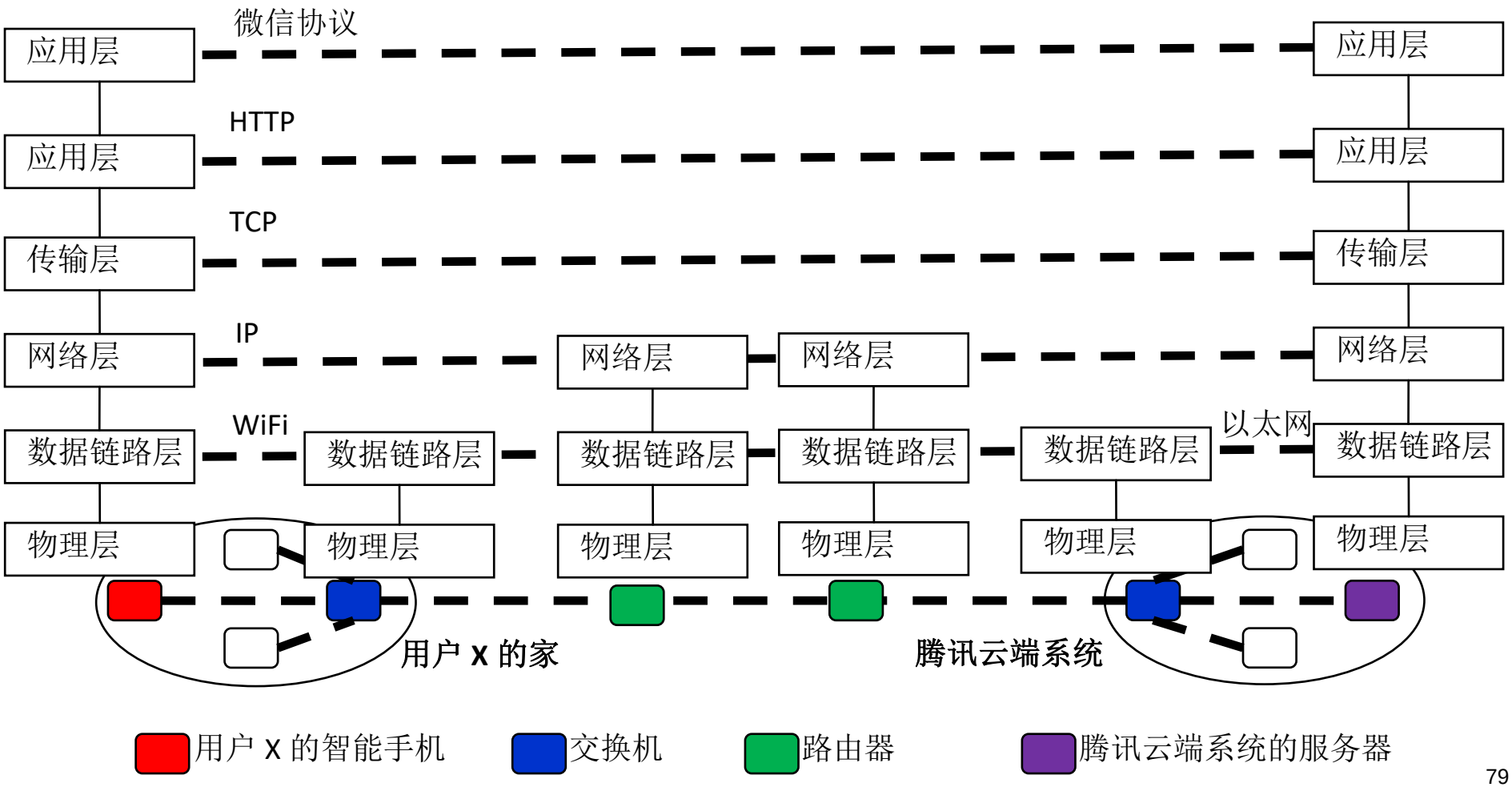
- 不能使用集中式的总线仲裁方式
- 1973年，麦特考夫发明以太网
  - 解决冲突的指数退避方法
    - 第一次传输试图失败后，等候[0, T]中间的一个随机值
    - 第二次重试失败后，等候[0, 2T]中间的一个随机值
    - 第三次重试失败后，等候[0, 4T]中间的一个随机值



以太网和四台电脑构成的局域网

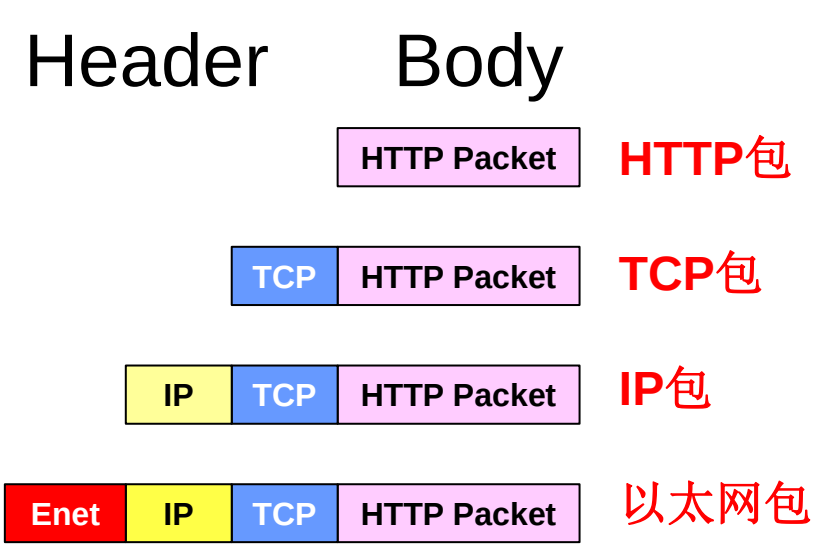
通信过程涉及互联网协议栈的哪些接口？

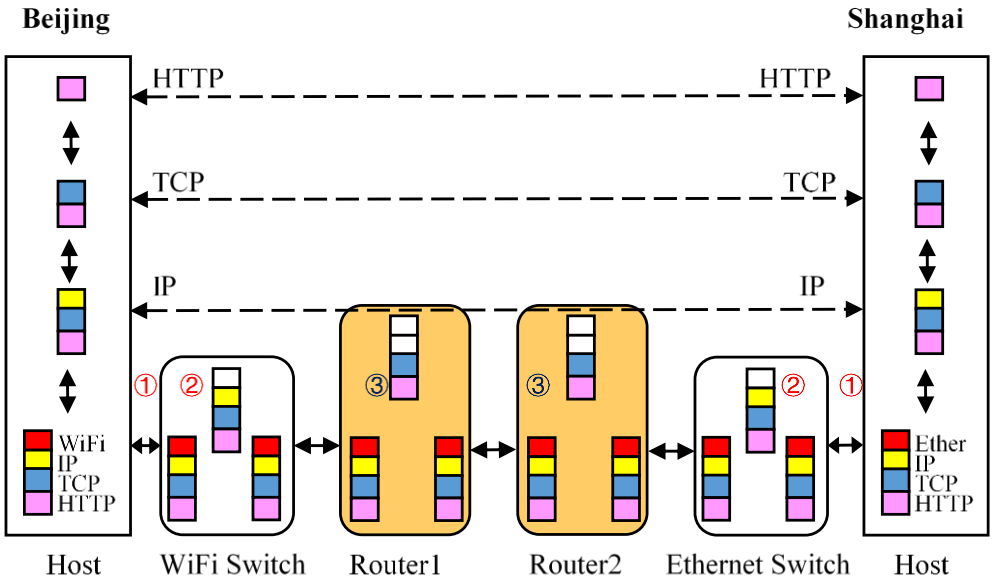
- 对等接口、层间接口



Can one send an upper layer packet without also sending a lower layer packet?

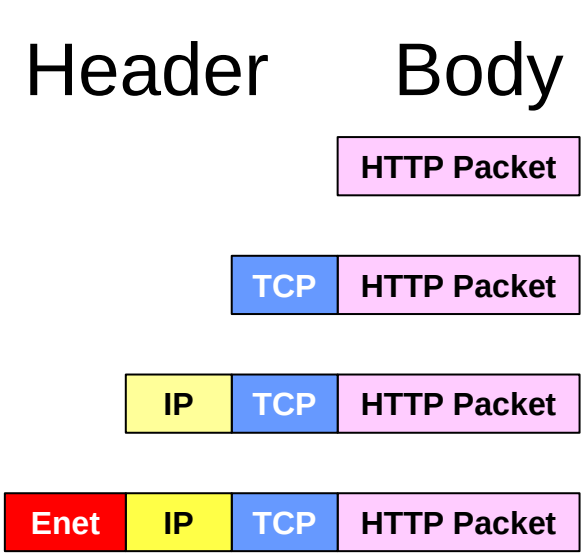
- Can the Web server in Shanghai send an HTTP packet to Zhang's Web browser in Beijing, without also sending a data link layer packet, e.g., an Ethernet frame?
- No! 不能只传上层数据包(如TCP包)，而不传下层包(IP包、以太网包)
  - Any information at the HTTP layer is wrapped in a data link layer packet, and eventually wrapped in a physical layer packet
  - One cannot send a high layer packet without also sending a packet of every layer below
  - When a packet enters a network, it is in a data link layer format and travels as wired and/or wireless signals

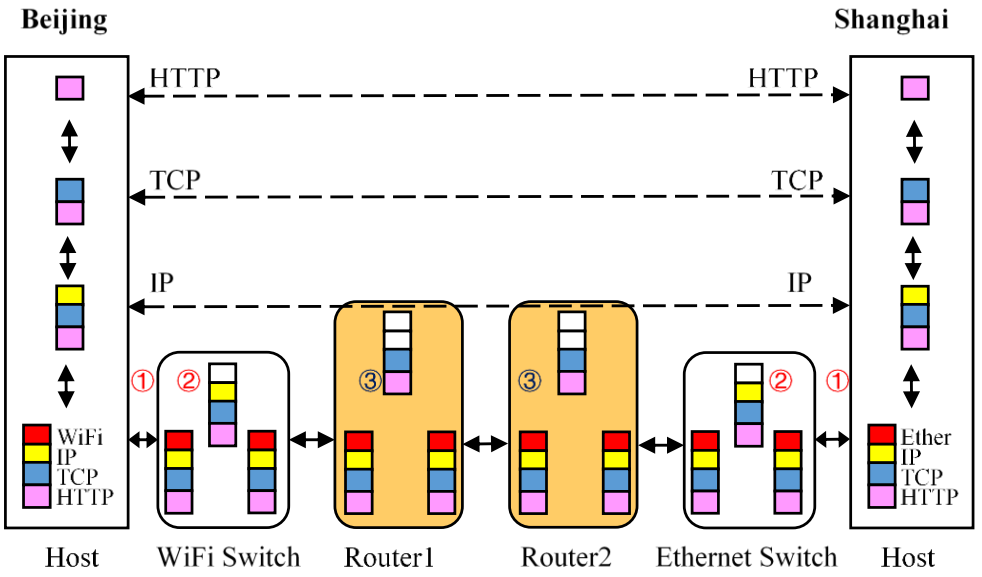




What is actually sent over the network hardware?

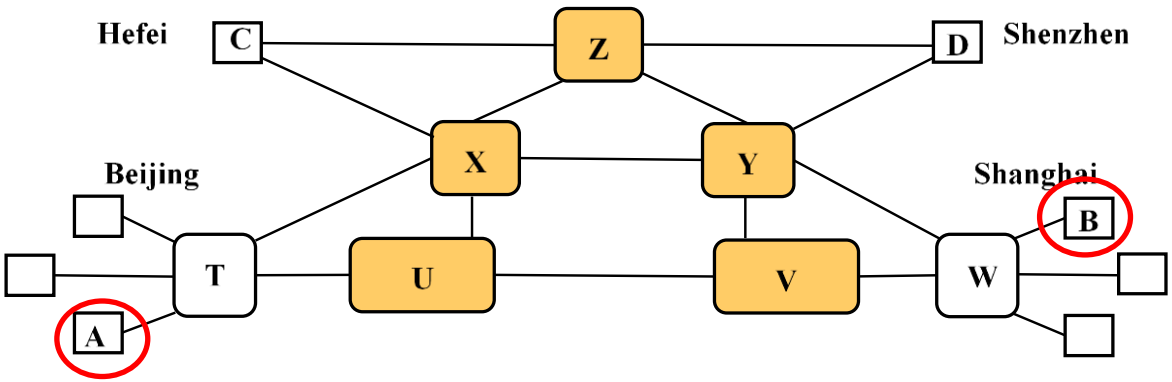
- Bit string of 0's and 1's
  任何数据包最终在物理层作为比特传递，即一串0或1信号
- Any packet is eventually encapsulated as one or more physical layer packets, which travel as wired or wireless signals
  - A physical layer packet is sent through electrical cables, electromagnetic waveforms or optical fibers, in a bit string of 0's and 1's
  - A 0 may be represented as a LOW voltage pulse or a LIGHTOFF state, while a 1 may be represented as a HIGH voltage pulse or a LIGHTON state





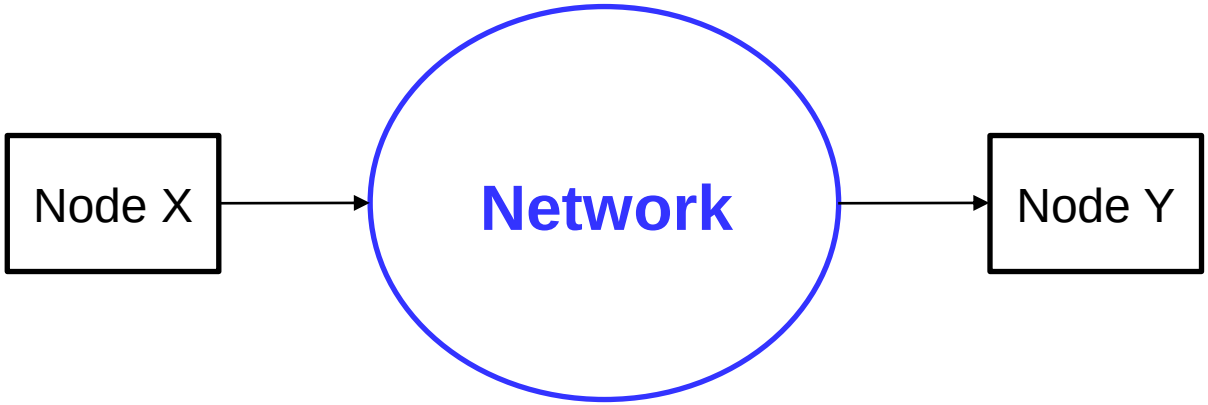
Do all packets travel through the same physical path?

- A message is sent from host A to host B
  - Do all packets of the message travel through the same physical path from host A to host B?
  - 从**A**到**B**的一条消息的数据包必然通过同一条通路吗？
  - Not necessarily. Internet has built-in redundancy 不一定。互联网有冗余通路
    - Possible physical paths for a 99-packet message from A to B
      - 1st packet of the message travels along the physical path **A-T-X-Y-W-B**
      - 49th packet traverses path **A-T-U-V-W-B**
        - Arriving at B before 1st packet
      - 99th packet traverses path **A-T-X-Z-Y-W-B**
    - Complete message is reassembled from the packets by their numbers



Latency and bandwidth

- We focus on one node sending a message to another node over a network 考虑最简单的网络
  - Node X sends a message of *m* bytes to node Y
    - What is the total time t to transmit the message?
- Hockney's formula: $t = t_0 + m/r_\infty$
  - Extreme values 极端值
    - 最小延迟 Minimal latency: t_0; 最大带宽 maximal bandwidth: r_∞
  - User experienced values 用户体验值
    - User experienced latency: t; User experienced bandwidth m/t



Compression 数据压缩

- Data compression: Technique to reduce file size
  - To save storage space and transmission time
- Lossless compression 无损压缩
  - Reduce file size without losing information
    - > gzip fib-10 (2011793 bytes)
    - To obtain a compressed file fib-10.gz (709090 bytes)
    - > gzip Autumn.bmp (9144630 bytes)
    - The compressed file is Autumn.bmp.gz (8224455 bytes)
  - Original file can be recovered from compressed file
    - > gzip -d Autumn.bmp.gz
- Lossy compression 有损压缩
  - Reduce file size while losing information
  - Original file cannot be recovered from compressed file

Network effect

- Network laws

  n = number of nodes

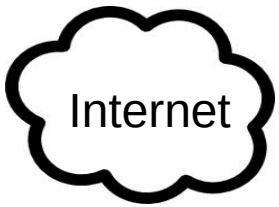
  - **Metcalfe's law** ($V \propto n^2$) 梅特卡夫定律
    - Value *V* of a network of *n* nodes is proportional to n^2
  - **Reed's law** ($V \propto 2^n$) 里德定律
    - Value *V* of a network of *n* nodes can scale exponentially with *n*, because the network can form 2^n subgroups
- Viral marketing 病毒市场现象
  - Markets grow wide and fast, like biologic viruses
  - Why?
    - Connected and 0-cost
      - Zero purchasing price
      - Zero usage cost
      - Zero propagation cost

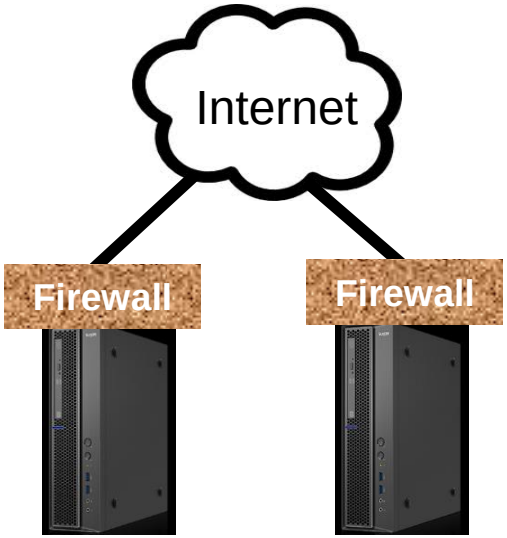
Cybersecurity issues

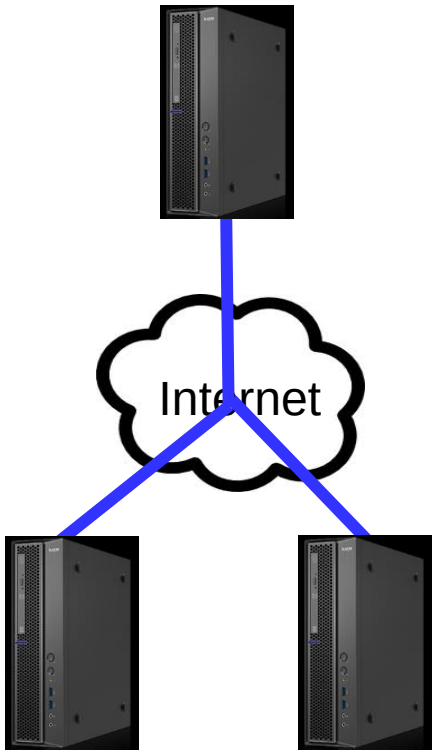
- Cybersecurity problems involve hardware, software and people 网络空间安全涉及硬件、软件、人
  - Not only software such as computer viruses，尽管软件似乎更明显
- Cyber attack types 存在多种攻击，不都是软件侵入
  - Malware: malicious software enabling an attacker to damage or gain unauthorized access to a computer 恶意软件
    - Computer viruses, worms, Trojan horses and spyware 病毒、蠕虫、木马、间谍软件
  - An attack does not have to be in a software form
    - Hardware exploitation 利用硬件的攻击
      - Meltdown: exploiting “out-of-order execution”, a feature of processor hardware 利用硬件的乱序执行
        - Enabling an attacker to read privileged information passwords
  - An attack does not have to install anything on the targeted system
    - Denial-of-service (DoS) attacks, distributed denial-of-service (DDoS) attack 拒绝服务攻击
    - Spams: unwanted emails 垃圾邮件
    - *Phishing*: phishing websites or phishing emails 钓鱼邮件、钓鱼网站

Counter measures

- Physical isolation: core computing systems disconnected from the Internet
- Firewalls: block or filter out undesirable messages
- Virtual private networks (VPNs)
- Antivirus software: detect and kill computer viruses 防病毒软件、杀毒软件
- 重要学术基础：Cryptography 密码学
  - Encryption: plaintext → ciphertext HELLO → KHOOR 加密：明文→密文
  - Decryption: ciphertext → plaintext KHOOR → HELLO 解密：密文→明文







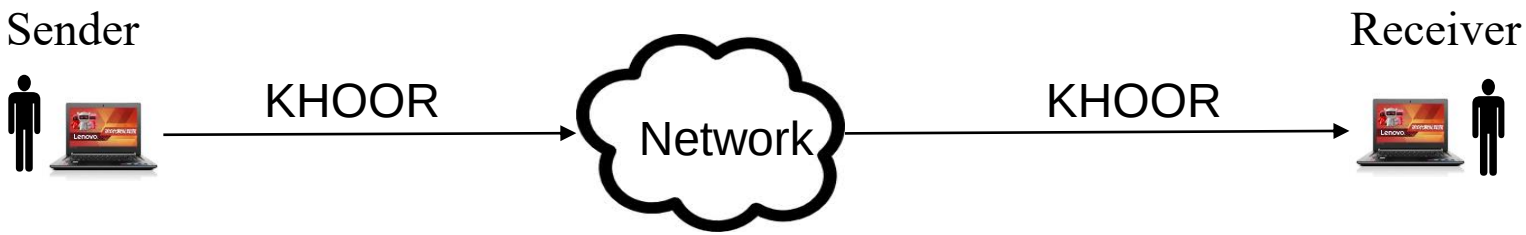
Symmetric-key encryption: Caesar cipher

对称加密：发送方与接收方共享秘钥；凯撒密码

- Sender and receiver share a key (3 in this example) 此例中秘钥是3
  - **Only a single key is used by both parties, thus symmetric**
- Sender encrypts the plaintext (string of capital letters)
  - By shifting each letter L 3 positions down the alphabet, i.e., ASCII(L)+3
  - E.g., 'H'+3 = 72+3 = 75 = 'K'

  and sends the ciphertext over the network to the receiver
- Receiver decrypts the ciphertext
  - By shifting each letter L up 3 positions, i.e., ASCII(L)-3
  - E.g., 'K'-3 = 75-3 = 72 = 'H'

Alphabet
A 65
B 66
C 67
D 68
E 69
F 70
G 71
H 72
I 73
J 74
K 75
…



HELLO → KHOOR
Shift 3 positions down
Encryption

KHOOR → HELLO
Shift 3 positions up
Decryption

6.2 Privacy issues 隐私

- Privacy: keeping a user's identity and personally identifiable information (PII) *private*.
- Personal information 个人信息=自然人信息
  - Any information relates to a natural person's identity
  - Includes personally identifiable information (PII)
    可区分、追溯到自然人的信息
  - Does not include anonymized personal information
- Personal information is broad
  - Such as personal names, ID numbers, personal photos or videos, website clicks records, voice signals, financial records, medical data
- Personal data can be revealed by technology
  - Utilizing metadata, data mining, AI
    哪怕没有直接暴露隐私信息，通过信息技术可计算出隐私信息

6.3 Professional norms

- ACM code of conduct: seven principles
  国际计算机协会行为规范的七原则
  - Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.
    为社会和人类的幸福做出贡献，承认所有人都是计算的利益相关者
  - Avoid harm. 避免伤害
  - Be honest and trustworthy. 诚实可靠
  - Be fair and take action not to discriminate. 做事公平，采取行动无歧视
  - Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.
    尊重他人工作，该工作产生新想法、新发明、创造性作品和计算工件
  - Respect privacy. 尊重隐私
  - Honor confidentiality. 尊重保密协议

https://www.acm.org/code-of-ethics

Form your own thoughtful judgement
了解ACM规范，形成自己的判断

- Understand the ACM code of conduct
  - You don't have to agree to it completely
    - The ACM code itself is evolving
  - But should try to understand what it says
- Apply it to the three examples in textbook, and form your own thoughtful judgement
  应用到教科书三个例子：Examples 67, 68, 69
  - Free flow versus professionally sharing of scientific data
    - 科学数据应该**自由流动**还是**专业性分享**？
      全球共享流感数据倡议组织（GISAID）的规范
  - Full disclosure versus responsible disclosure
    - 组织内部出现可能有害社会的漏洞，应该向社会完全曝光还是负责任地通报
  - The case of the Morris worm
    - 善意的（无恶意的）科学研究工作是否可以越界？

谢谢 Thank You

Q&A

zxu@ict.ac.cn

中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY