计算思维与计算科学

计算的边界

计算的边界

可计算性与计算的等效性

计算的边界

- 计算模型
  - 有限状态机
  - 递归函数
  - 图灵机
- 计算模型的等效性
- 可计算性与停机问题

计算模型：等效性和边界

- 我们设计的指令系统能否实现所有编程所需要的功能?
- 不同的指令系统之间是否等价?
- 解决一个计算问题需要哪些基本操作?
- 是否有其他的计算模型?

- 我们的计算模型能否解决所有的问题?

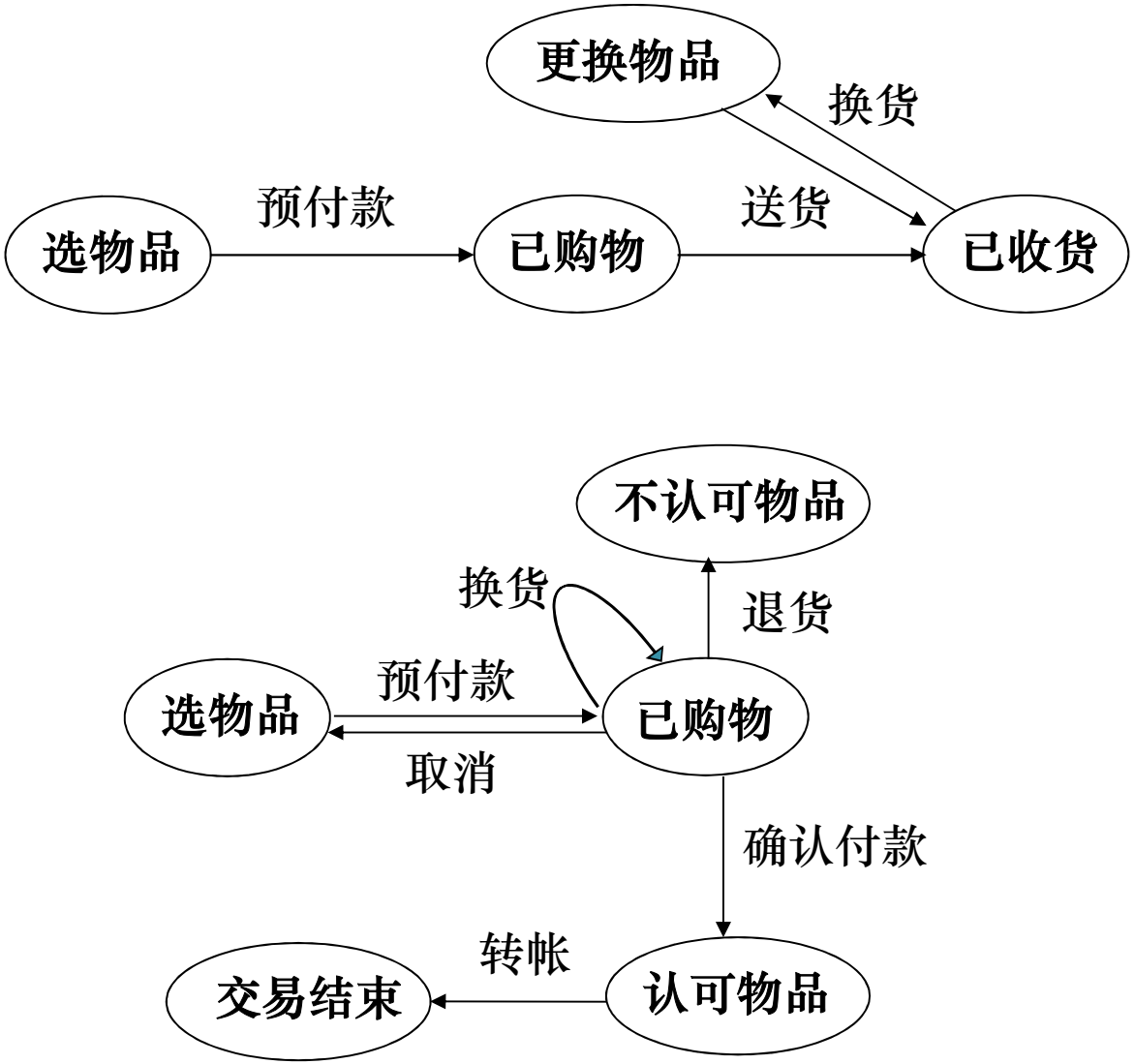
计算能力的理论探索

- 什么样的问题是可以得到答案的?
- 希尔伯特23个问题
  - 2.算术公理之相容性
  - 10.不定方程可解性

- 计算的模型
  - 时序逻辑电路
  - 有限自动机
  - 递归函数
  - Lambda演算
  - 图灵机

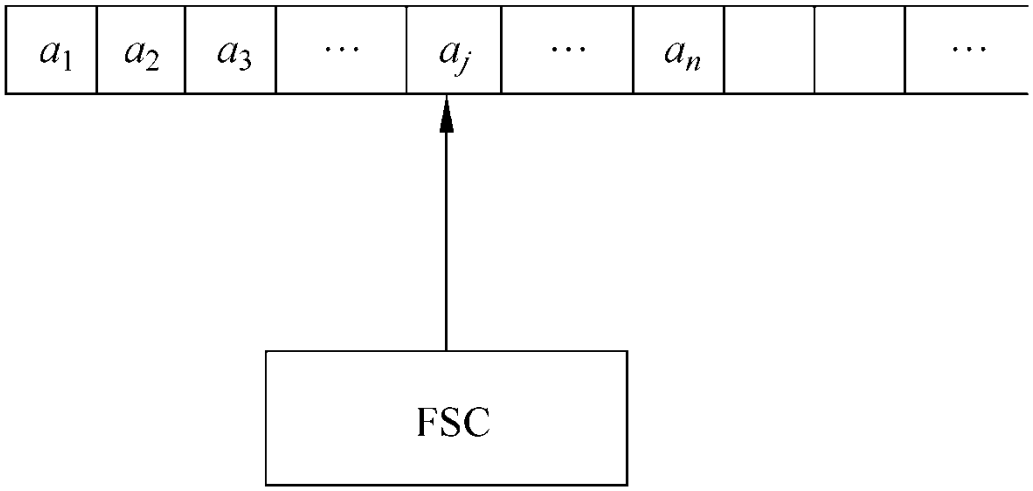
有限状态系统——淘宝网上购物

- 顾客、店家和支付宝网三方之间的交互限于以下几种事件：
  1、顾客告诉店家购买某种物品，决定预付款购物。 并将钱款转入支付宝。
  2、顾客决定取消预付款。 顾客通知支付宝，把购物这笔钱保留在自己的银行帐号上。
  3、店家送货给顾客。
  4、顾客收到货后
     （1）确认付款。 通知支付宝将钱款划拨到店家帐号， 转到（5）
     （2）退货。把购物这笔钱保留在自己的银行帐号上，结束。
     （3）换货。寄回店家，店家重送货给顾客。
  5、支付宝将这笔钱转帐。 把顾客购物这笔钱划拨给店家的帐号。
- 以上的事件以及事件间在一定条件下转化的情况，可以表示成有限状态系统，每个状态表示某一方所处的局面。

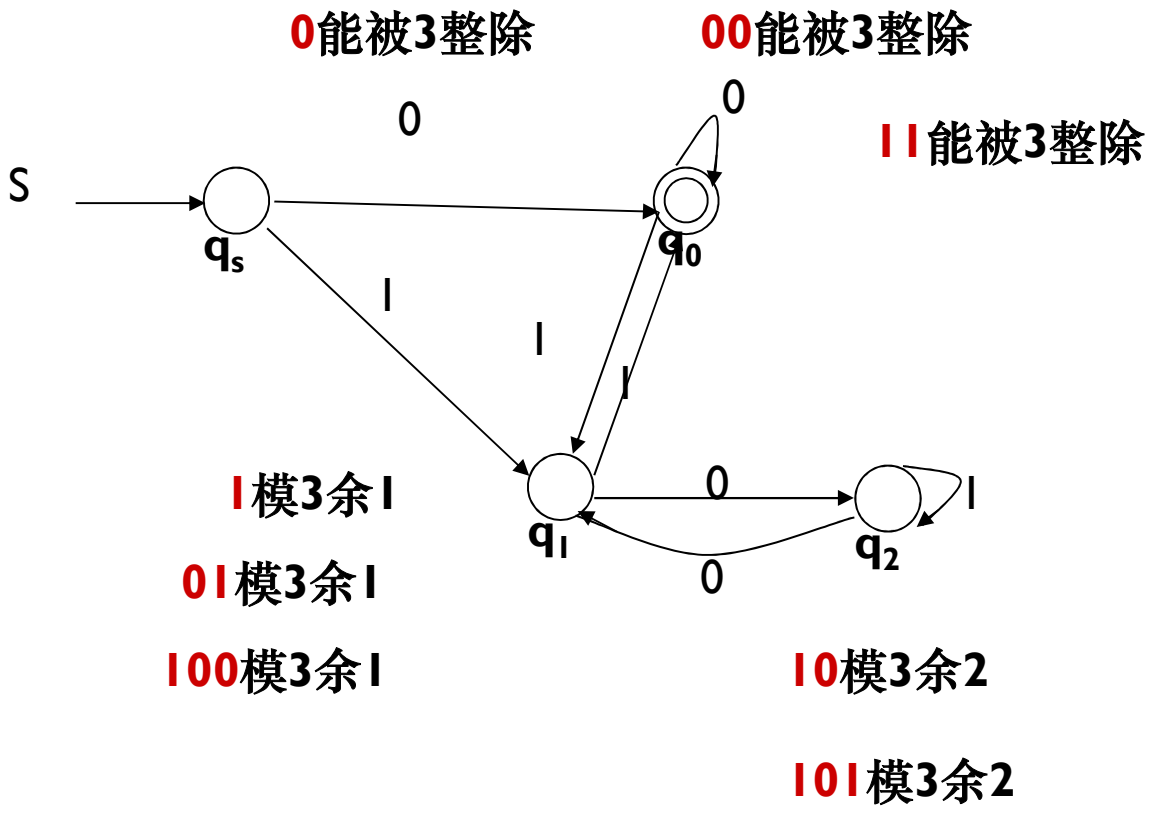

更换物品
换货
预付款
送货
选物品
已购物
已收货
不认可物品
换货
退货
预付款
选物品
已购物
取消
确认付款
转帐
交易结束
认可物品

有限自动机DFA

- 物理模型
  - 一个右端无穷的输入带。
  - 一个有限状态控制器(finite state control,FSC) 。
  - 一个读头。
- 系统的每一个动作由三个节拍构成：
  - 读入读头正注视的字符；
  - 根据当前状态和读入的字符改变有限控制器的状态；
  - 将读头向右移动一格。



- 例 构造一个DFA，它接受的语言为{x|x∈{0，1}*，且当把x看成二进制数时，x要能被3整除
- 分析：
- 当二进制数x的位数向右不断增加时，它的值的增加很有规律：x0的值等于2x，x1的值等于2x+1。
- 经这样分析以后，FAM除初始状态外，只需要设3个状态：模3余0状态，模3余1状态，模3余2状态。
- 同时，模3余0状态也是终结状态


0能被3整除
00能被3整除
11能被3整除
S
q_s
q_0
q_1
q_2
0
0
1
1
1
0
1
0
1模3余1
01模3余1
100模3余1
10模3余2
101模3余2

有限自动机的能力

- 能被确定有限状态自动机识别的语言是正则语言。
- 对一个确定有限状态自动机，下述判定问题都可以判定，并且存在有效的算法。
  - 该自动机识别的语言是否为空集。
  - 该自动机识别的语言是否为有限集。
  - 该自动机是否与另一个确定有限状态自动机识别同一个的语言。
- 有限自动机是一种十分重要的时序逻辑电路设计模式
- 真实（内存有限）的计算机可以看作是一种有限自动机

有限自动机的限制

- 很多简单的语言，包括需要多于恒定空间来解决的任何问题，不能被 DFA 识别
  - 正确配对的括号组成的语言，比如 (()())。
  - 由形如 $a^n b^n$ 的字符串组成的语言，就是有限数目个 a，随后是相等数目个 b。可以证明没有 DFA 有足够状态来识别这种语言（通俗地说，因为需要至少 2n 个状态，而 n 是不恒定的）。

自动机

- 非确定性自动机
- 下推自动机
- 线性带限自动机

- 有限自动机与正则表达式

递归函数

- 通过一些算子将最基本的函数构造成复杂函数
- 基本函数：
  - 零函数$0(x)=0$（其值恒为0）；
  - 射影函数$L_{m,n}(x_1,x_2\ldots x_m)=x_n$；
  - 后继函数$S(x)=x+1$
- 算子
  - 复合算子
  - 递归
  - 求逆

$$\begin{cases} f(u_1,u_2,\cdots,u_n,0)=g(u_1,u_2,\cdots,u_n), \\ f(u_1,u_2,\cdots,u_n,S(x)) \\ \quad =h(u_1,u_2,\cdots,u_n,x,\ f(u_1,u_2,\cdots,u_n,x)) 。 \end{cases}$$

$$\begin{cases} f(u,0)=g(u) \\ f(u,S(x))=h(u,x,f(u,x)) 。 \end{cases}$$

$$\begin{cases} f(u,0)=g(u) \\ f(u,x)=h(u,P(x),f(u,P(x)) \end{cases}$$

μ算子（无界查找算子）

- 用 ***x*** 表示任意多个参数
- μ算子的输入是一个函数f(y,***x***)，并返回一个函数，它的参数是 ***x***。
- 这个函数 μy*f* 返回最小的自然数y，使得 f(y,***x***) 结果为0。如果这样的y不存在，则μy *f*是对特定参数 ***x*** 是未定义的。

递归函数举例

- 加法、乘法
- 除法（μ算子）
- 布尔化、逻辑运算、iff函数

```
def to_bool(x:Int) : Int= {
  if (x==0) 0 else 1;
}
def neg(x:Int) : Int = {
  if (x==0) 1 else 0;
}
def iff(b:Int, x:Int, y:Int ) : Int = {
  to_bool(b) * x + neg(b) * y
}
def mod(x:Int, y:Int) :Int = {
  x - (x / y) * y
}
def is_factor(x:Int, y:Int) :Int = {
  neg(mod(x,y))
}
def check(n:Int, m:Int) :Int = {
  if (m > 1) iff(is_factor(n,m), 1, check(n, m-1))
else 0
}
def is_prime(n:Int) :Int = {
  neg(check(n, n-1))
}
```

递归函数

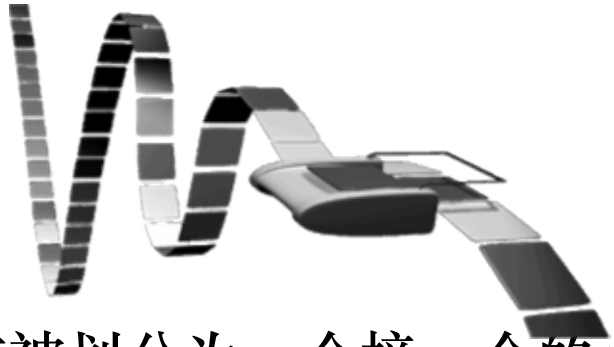
- 通常的函数都是可以用（一般）递归函数表示的
- 普通程序设计语言中的循环可以用尾递归表示
- 递归函数的计算能力相当于普通的数学表示
- 阿克曼

$$A(m,n)=$$

Ackermann(0,n)=n+1
Ackermann(1,n)=n+2
Ackermann(2,n)=2*n+3
Ackermann(3,n)=2^(n+3)-3
Ackermann(4,n)=2^2^2^……^2-3，乘幂中共有n+3个2。

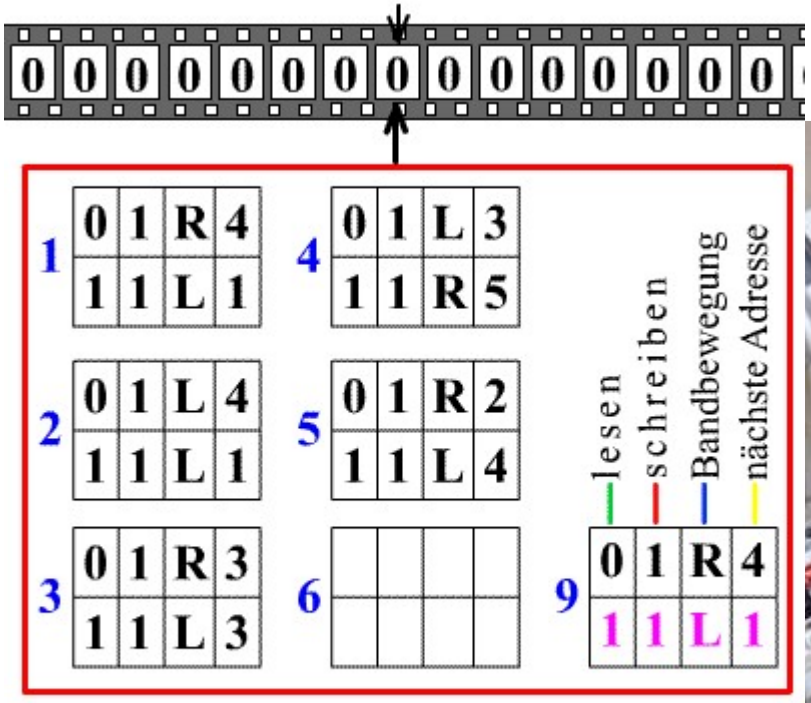
当m≥4，Ackermann函数的增长快得惊人。
Ackermann(4,0)=13，Ackermann(4,1)=65533，
Ackermann(4,2)=2^65536-3有19729位，
而Ackermann(4,3)则即使是位数也不易估计。
Ackermann(5,0)=65533，
Ackermann(5,1)=Ackermann(4,65533)

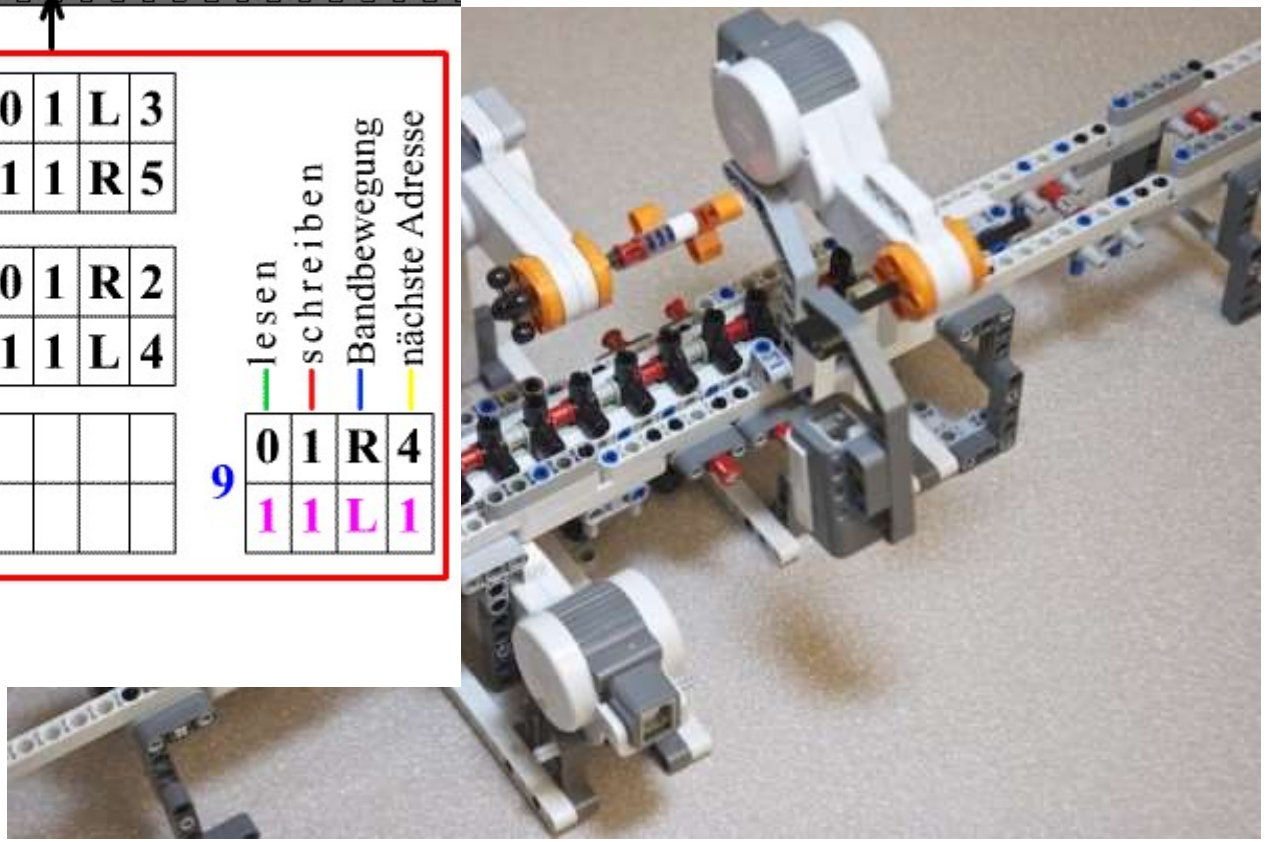
图灵机



- 一条无限长的纸带TAPE。纸带被划分为一个接一个的小格子，每个格子上包含一个来自有限字母表的符号，字母表中有一个特殊的符号□表示空白。纸带上的格子从左到右依次被编号为0, 1, 2, ...，纸带的右端可以无限伸展。
- 一个读写头HEAD。该读写头可以在纸带上左右移动，它能读出当前所指的格子上的符号，并能改变当前格子上的符号。
- 一套有限的控制规则TABLE。它根据当前机器所处的状态以及当前读写头所指的格子上的符号来确定读写头下一步的动作，并改变状态寄存器的值，令机器进入一个新的状态。
- 一个状态寄存器。它用来保存图灵机当前所处的状态。图灵机的所有可能状态的数目是有限的，并且有一个特殊的状态，称为停机状态。

图灵机-图片


0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 R 4
1 1 1 L 1
2 0 1 L 4
1 1 L 1
3 0 1 R 3
1 1 L 3
4 0 1 L 3
1 1 R 5
5 0 1 R 2
1 1 L 4
6
9 0 1 R 4
1 1 L 1
lesen
schreiben
Bandbewegung
nächste Adresse



图灵机的编程举例（修改）

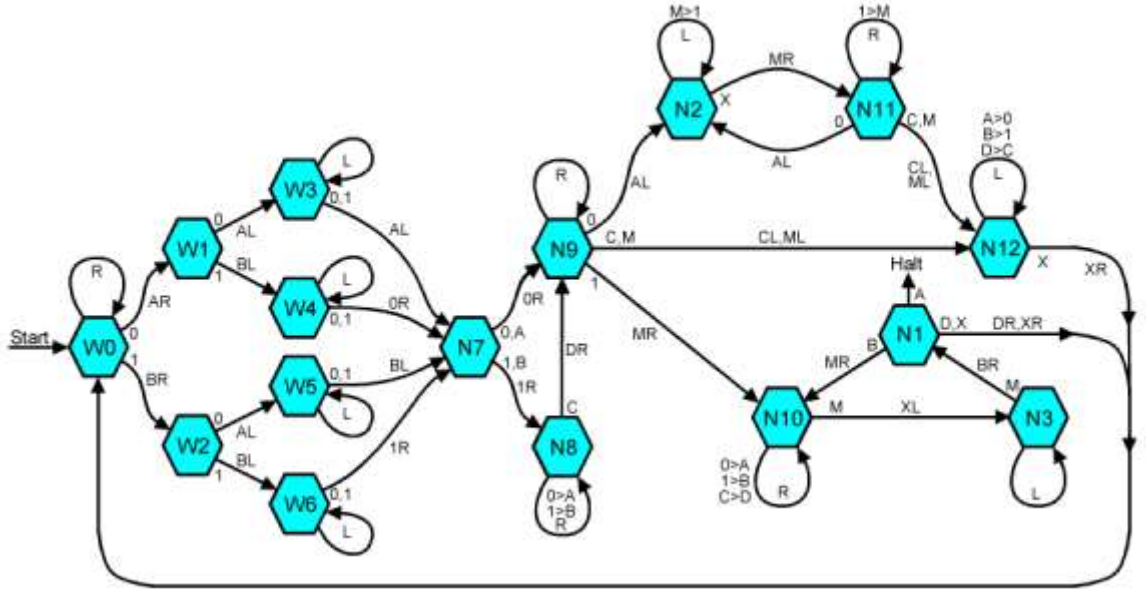
- 图灵机编程：
  - 1进制加法
  - 二进制加1
  - 判断形如 $a^n b^n$ 的字符串
  - 复制数据
  - 二进制加减法
- 编程举例：二进制加法

扩充图灵机

- 图灵机变形
  - 扩充符号集
  - 改变读写规则
  - 增加局部存储
  - 增加纸带，甚至是变成一个棋盘
- 可以证明，这些变形后的图灵机，都可以使用原始图灵机进行模拟，因此计算能力上是彼此等价的

通用图灵机

- 能够读取其他图灵机的程序，并模拟运行的图灵机
- 将某个图灵机程序进行编码，通用图灵机以此作为输入，就可以完成该程序的模拟执行



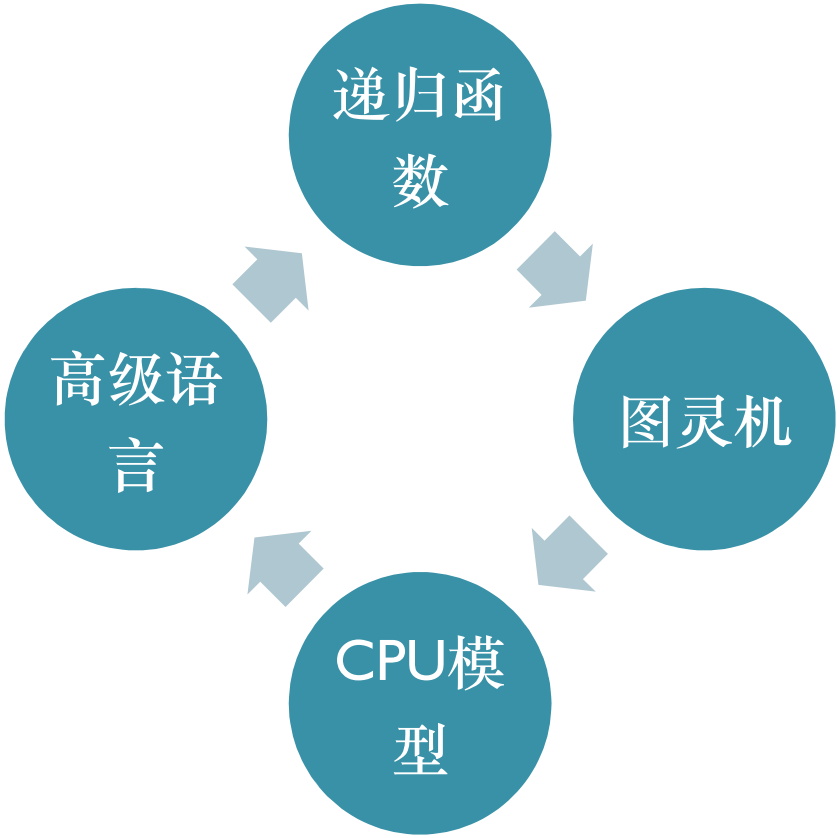
丘奇-图灵命题

- 递归函数、图灵机、λ演算在计算能力上是等价的
- 可计算函数：能够使用图灵机（或其等价模型）计算的函数

- 这意味着：
  - 图灵机是计算能力完备的计算模型
  - 不是所有问题都能使用上述模型来计算（解决）的

计算模型的等效性

- 使用A模型模拟B模型
- 使用A模型，编写“解释程序”
  - 将B模型的“程序”和处理的数据作为输入
  - 将B模型的每一步基础操作在A模型下模拟实现
  - 按照B模型的执行流程，依次执行程序
  - 按照B模型，决定是否终止运行



计算模型的相互模拟

- 递归函数模拟图灵机
  - 将图灵机的纸带输入t，状态s，当前读写头位置p看做递归函数的变量fun turing(t,s,p,prog)
  - 如果s是终止状态，则返回
  - 检查纸带t的位置p的值x
  - 根据x和s找出对应的图灵机指令
  - 根据指令，改变t,s,p
  - 递归调用turing

计算模型的相互模拟

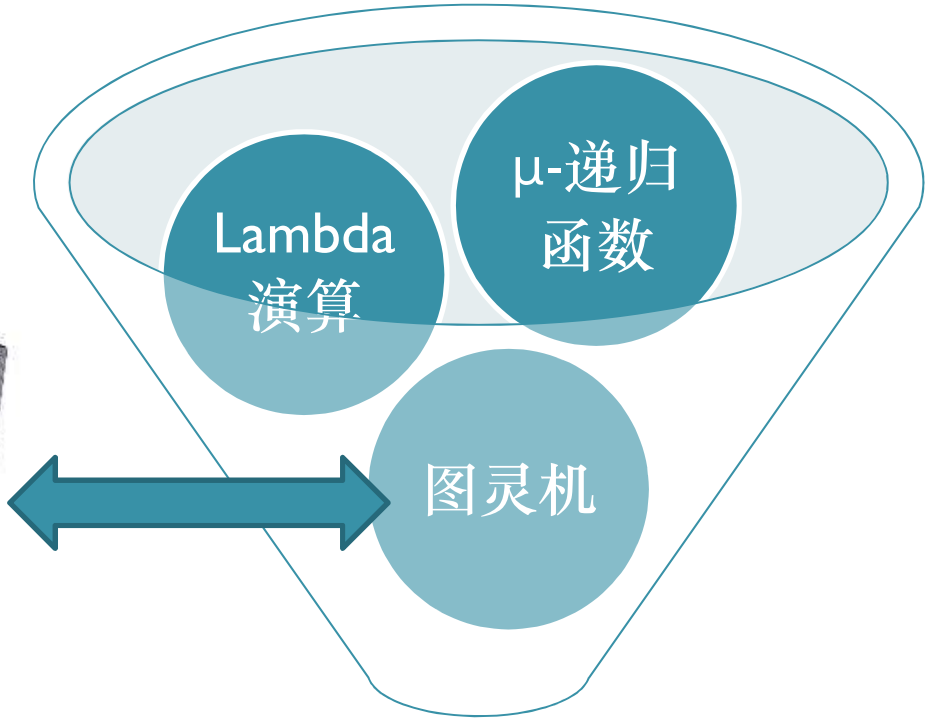
- 图灵机模拟CPU
  - 将CPU中的内部寄存器（指令计数器PC、累加器A、状态寄存器F）在纸带上以固定位置保存，定义临时变量，剩下纸带模拟内存。
  - CPU的执行流程以状态方式模拟
    - 取指令到临时变量c
    - 根据指令的编码进行分支
    - 内存取操作数到临时变量x
    - 运算，结果
    - 保存数据到内存
    - 如跳转指令则改变PC

计算模型的相互模拟

- CPU模型实现高级语言
- 高级语言实现递归函数
- 高级语言实现图灵机

- 计算机语言的图灵完备性和等效性

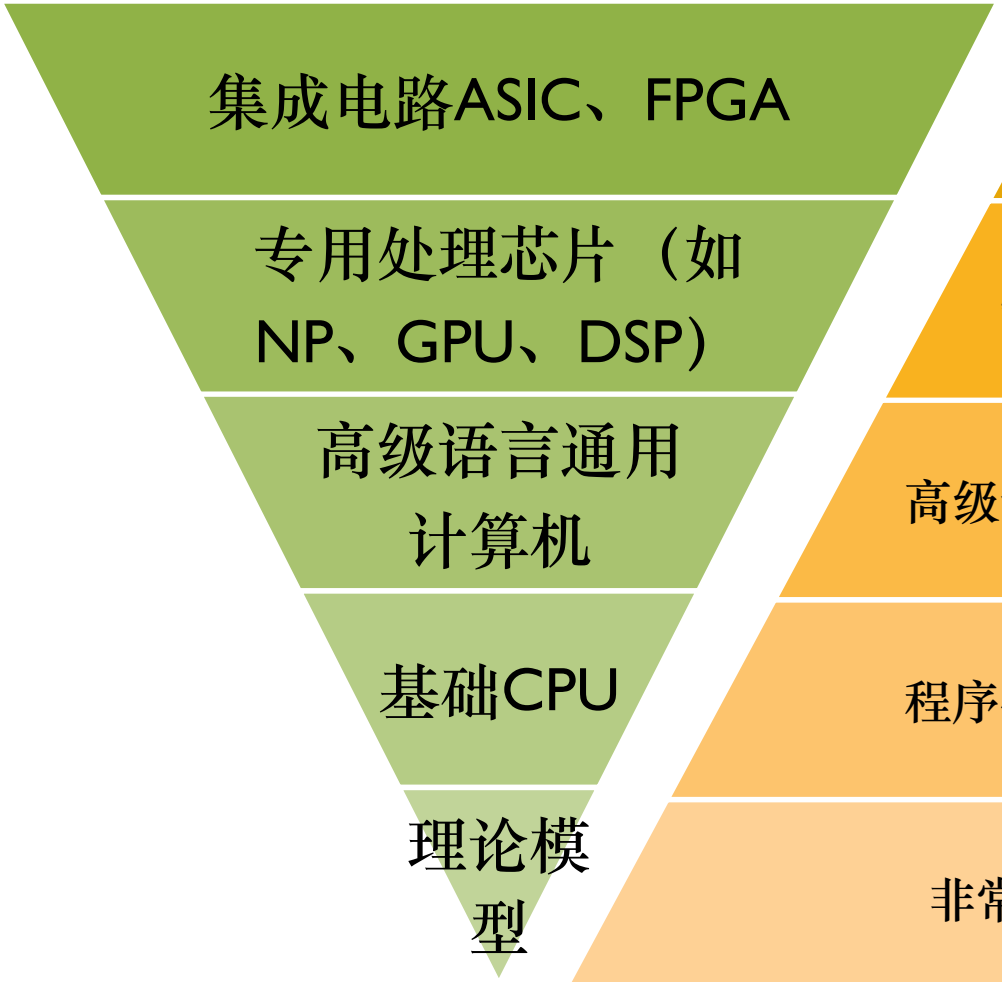
计算能力的等价性

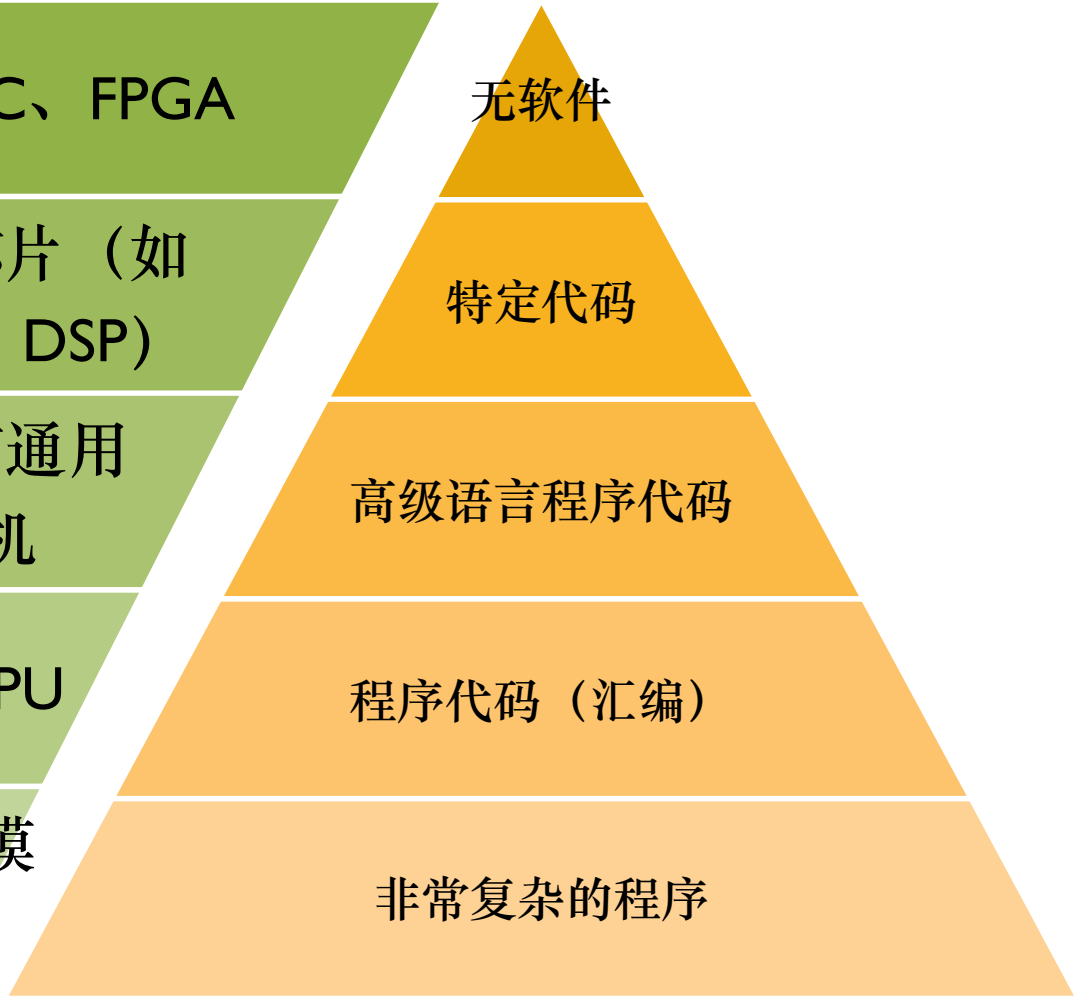


无限内存的计算机模型和图灵机是等价的

可计算的函数

软件-硬件的等效性


集成电路ASIC、FPGA
专用处理芯片（如NP、GPU、DSP）
高级语言通用计算机
基础CPU
理论模型


无软件
特定代码
高级语言程序代码
程序代码（汇编）
非常复杂的程序

不可计算函数

- 所有图灵机程序构成可数集合
- 所有可能的判定函数（定义域是自然数，值域是[0,1]），是不可数集合。
- 因此，存在不可计算的函数

停机问题

- 定义 停机函数$H(x, y)$: $N \times N \rightarrow \{0, 1\}$
- $H(x,y) = \begin{cases} 1 \ when \ \varphi_x(y)\text{停机} \\ 0 \quad otherwise \end{cases}$
- 如果停机问题可解，则
  - 哥德巴赫猜想可证
  - 大部分数论猜想都可证

Gödel不完备性定理

- 2002年美国《时代周刊》列出“20世纪震撼人类思想界的四大伟人”：
- 爱因斯坦（Albert Einstein）
- 图灵（Alan Turing）
- 哥德尔（Kurt Gödel）
- 凯恩斯（John Keynes）

Kurt Gödel, 1906—1978

Hilbert问题

- 1900年巴黎数学家会议上，希尔伯特遵从“世界上没有不可知”，“人类理性提出的问题人类理性一定能够回答”的哲学信念，提出23个数学问题，其中的第二个问题就是建立整个数学的一致性（即无矛盾性或称协调性）。
- 1920年代希尔伯特本人曾提出了一个使用有穷方法建立实数和分析的一致性的方案，称为希尔伯特元数学方案。

Gödel不完备性定理

- 第一定理
- 任意一个包含一阶谓词逻辑与初等数论的形式系统，都存在一个命题，它在这个系统中既不能被证明也不能被否定。

- 第二定理
- 如果系统S含有初等数论，当S无矛盾时，它的无矛盾性不可能在S内证明。

课后作业

- 一、使用C语言，不得使用if以外的控制流语句，不使用临时变量，使用递归方法，完成以下函数：
  - 1、输入正整数n和m，判断n的所有因数（包括1，不包括n）之和是否与n相差m。
  - 2、给定一个整数数组（长度为n），找出其中长度最长的连续递增序列的长度。
  - 3、讨论问题2的算法的复杂度，并尽量优化算法。
- 二、使用图灵机模型，完成以下程序：
  - 1、判断纸带上的01串（以符号B开头和结尾）是否左右对称。
  - 2、将纸带上起始位置开始的二进制数加1，二进制数的最低位在最左方。纸带上只有两个符号a和b，每个二进制数位用两个符号表示，aa表示0,ab表示1，ba和bb表示数的结尾。

延伸阅读

- 科学松鼠会：计算的极限
- 人列计算机
- 计算能力的发展-摩尔定律

课后报告

- 归纳出C语言的最简化内容
- 计算机硬件发展
  - 微机CPU的发展史
  - 摩尔定律及其适用范围
  - 国内CPU的发展
- 编写一个图灵机的解释程序