计算系统与程序

---程序的作用和本质

战德臣

哈尔滨工业大学　　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

如何设计实现一个基本计算系统?

首先，设计并实现系统可以执行的基本动作(可实现的)，例如

- “与”动作
- “或”动作
- “非”动作
- “异或”动作

已知的基本事实是:
“加减乘除运算都可转换为加减法运算来实现”
“加减法运算又可以转换为逻辑运算来实现”

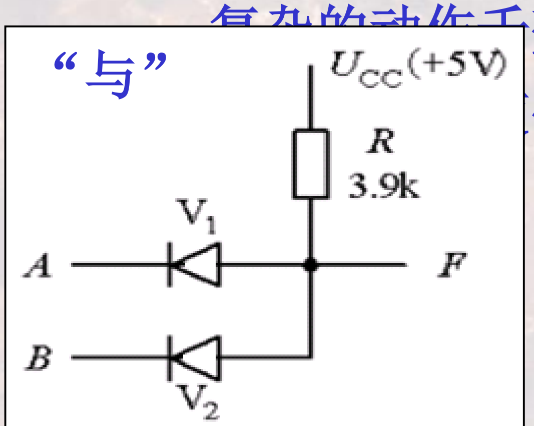
那么，复杂的动作呢?

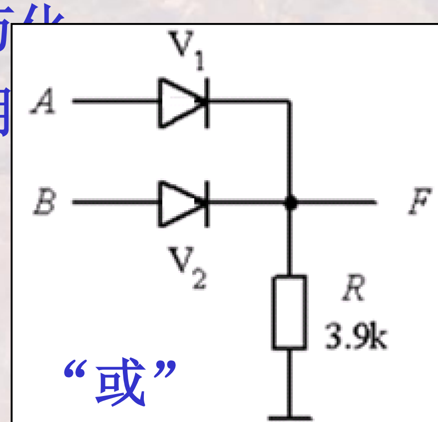
系统需要提供复杂的动作

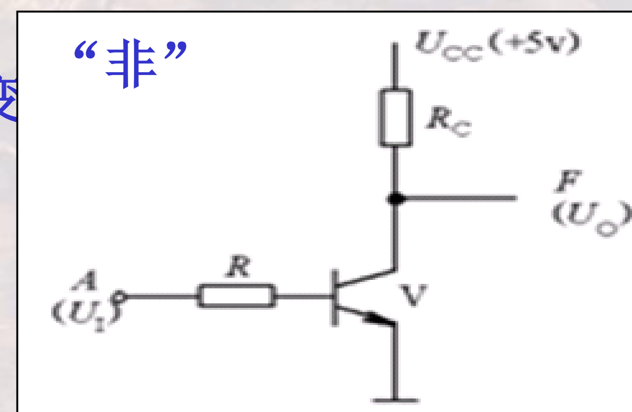
复杂的动作千变万化

使用

而变







如何设计实现一个基本计算系统?

程序:由基本动作指令构造的，若干指令的一个组合或一个执行序列，用以实现复杂动作

复杂动作

((A AND B) AND C) OR (NOT C))

拆解开

X= A AND B
X= X AND C
Y= NOT C
X= X OR Y

系统

“与”动
作
“或”动作
“非”动作

AND
OR
NOT

指令：控制基本动作执行的命令

如何设计实现一个基本的计算系统?

程序:由基本动作指令构造的,若干指令的一个组合或一个执行序列,用以实现复杂动作

复杂动作

((A AND B) AND C) OR (NOT C))

系统

“与”动作

“或”动作

“非”动作

AND

OR

NOT

自动解释程序中的各种组合,并按次序调用指令(基本动作)予以执行

程序执行机构

指令:控制基本动作执行的命令

计算系统与程序-程序的作用和本质

(4) 计算系统与程序?

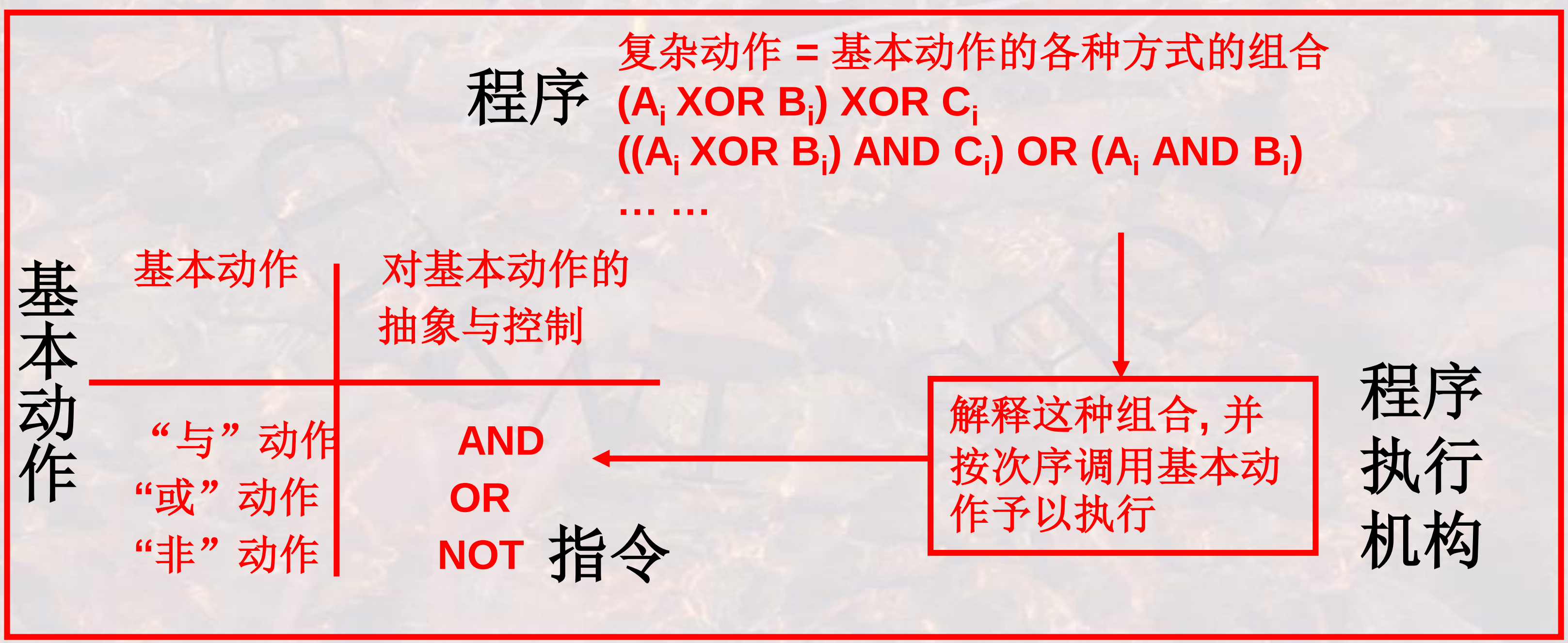
战德臣 教授

计算系统 = 基本动作 + 指令 + 程序执行机构

指令 = 对可执行基本动作的抽象，即控制基本动作执行的命令

程序 = 基本动作指令的一个组合或执行序列, 用以实现复杂的动作

程序执行机构 = 负责解释程序即解释指令之间组合，并按次序调用指令即调用基本动作执行的机构



计算系统与程序-程序的作用和本质
(5) 程序：组合-抽象-构造?

战德臣 教授

抽象： 将经常使用的、可由低层次系统实现的一些复杂动作，进行**命名**，以作为高层次系统的指令被使用

一种较高抽象层次的系统

基本动作	对基本动作的抽象与控制
“加”动作	+
“减”动作	-
“乘”动作	x
“除”动作	÷

指令

程序

复杂动作 = 基本动作的各种方式的组合

(V1 + V2) x (V3 ÷ V4) ÷ V5

(V1 ÷ (V2 x (V3 + V4)) - (V5 x V6)

… …

解释这种组合，并按次序调用基本动作予以执行

程序执行机构

抽象

一种较低抽象层次的系统

程序

复杂动作 = 基本动作的各种方式的组合

(A_i XOR B_i) XOR C_i

((A_i XOR B_i) AND C_i) OR (A_i AND B_i)

… …

基本动作

基本动作	对基本动作的抽象与控制
“与”动作	AND
“或”动作	OR
“非”动作	NOT

指令

解释这种组合，并按次序调用基本动作予以执行

程序执行机构

什么是程序?

程序的本质是什么?

组合 抽象 → 构造

运算式的组合-抽象与构造

---程序构造示例I-计算对象的定义-构造与计算

战德臣

哈尔滨工业大学　　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

由数值，到基本运算组合式

100

205

实际的数值

(100 + 205)

中缀表示法, 用运算符(即前述的指令)将两个数值组合起来，运算符在中间

(+ 100 205)

前缀表示法, 用运算符(即前述的指令)将两个数值组合起来，运算符在前面

将运算符表示的操作应用于后面的一组数值上，求出结果

(+ 100 205 300 400 51 304)

一个运算符可以表示连加，连减等情况，

由数值，到基本运算组合式

(+ 100 205)

(- 200 50)

(200 5)*

(20 5 4 2)*

(- 20 5 4 2)

(+ 20 5 4 2)

一起练习,书写程序, … …

运算组合式的“嵌套”及其计算过程

(+ 100 205)

(+ (+ 60 40) (- 305 100))

(* (* 3 (+ (* 2 4) (+ 3 5))) (+ (- 10 7) 6))

计算过程

(* (* 3 (+ (* 2 4) (+ 3 5))) (+ (- 10 7) 6))

(* (* 3 (+ 8 8)) (+ 3 6))

(* (* 3 16) 9)

(* 48 9)

432

命名计算对象和构造中使用名字及计算中以计算对象替换名字

(define height 2) —— *名字的定义：定义名字height与2关联，以后可以用height来表示2*
一种类型的名字：数值型的名字

(+ (+ height 40) (- 305 height)) —— *名字的使用*

(+ (* 50 height) (- 100 height))

注意：不同类型的对象可以有不同的定义方法。这里统一用define来表示，在具体的程序设计语言中是用不同的方法来定义的

运算式的组合-抽象与构造---程序构造示例

(3)如何用名字简化运算组合式的构造?--抽象

战德臣 教授

命名计算对象和构造中使用名字及计算中以计算对象替换名字

```
(define pi 3.14159)
(define radius 10)
(*  pi  (* radius radius))
```

```
(*  pi  (*  radius radius))
(*  pi  (* 10  10))
(*  pi  100)
(* 3.14159 100)
```

计算

```
(define  circumference  (* 2 p
```

```
(*  circmference  20)
```

```
(*  circmference  20)
(* (* 2  pi  radius) 20)
(* (* 2  3.14159  10) 20)
(* 62.8318  20)
1256.636
```

计算

运算式的组合-抽象与构造

---程序构造示例II-运算符的定义-构造-与计算

战德臣

哈尔滨工业大学　　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

战德臣 教授

命名新运算符和构造中使用新运算符及执行中以过程替换新运算符

(define (square x) (* x x))

x^2

名字的定义：定义名字square为一个新的运算，即过程或称函数

另一种类型的名字：运算符型的名字

新运算符，即过程名或函数名

形式参数，使用时将被实际参数所替代

过程体，用于表示新运算符的具体计算规则，其为关于形式参数x的一种计算组合。

(square 3)

(square 6)

名字的使用

注意：不同类型的对象可以有不同的定义方法。这里统一用define来表示，在具体的程序设计语言中是用不同的方法来定义的

命名新运算符和构造中使用新运算符及执行中以过程替换新运算符

(square 10) —————— *名字的使用*

(square (+ 2 8))

(square (square 3))

(square (square (+ 2 5)))

(define (SumOfSquare x y) (+ (square x) (square y)))

x^2+y^2

(SumOfSquare 3 4)

(+ (SumOfSquare 3 4) height)

命名新运算符和构造中使用新运算符及执行中以过程替换新运算符

(define (NewProc a) (SumOfSquare (+ a 1) (a 2)))*

$(a+1)^2+(a*2)^2$

(NewProc 3)

(NewProc (+ 3 1))

战德臣 教授

命名新运算符和构造中使用新运算符及执行中以过程替换新运算符

含名字的运算组合式的计算方法：求值、代入、计算

*(NewProc (+ 3 1))*的两种计算过程示意

```
(NewProc (+  3  1))
(NewProc 4)
(SumOfSquare (+  4  1) (*  4  2))
(SumOfSquare 5  8)
(+  (Square  5)  (Square  8))
(+  (*  5  5)  (*  8  8))
(+  25  64)
89
```

先求值，再代入

命名新运算符和构造中使用新运算符及执行中以过程替换新运算符

含名字的运算组合式的计算方法：代入、求值、计算

***(NewProc (+ 3   1))*的两种计算过程示意**

(NewProc (+ 3 1))	*先代入，后求值*
(SumOfSquare(+ (+ 3 1) 1) ((+ 3 1) 2))*	
(+ (Square (+ (+ 3 1) 1) (Square ((+ 3 1) 2)))*	
(+ ((+ (+ 3 1) 1) (+ (+ 3 1) 1)) (* (* (+ 3 1) 2) (* (+ 3 1) 2)))*	*代入阶段*
(+ ((+ 4 1) (+ 4 1)) (* (* 4 2) (* 4 2)))*	*求值阶段*
(+ (5 5) (* 8 8))*	
(+ 25 64)	
89	

运算式的组合-抽象与构造

---程序构造示例III-条件组合式的构造及总结

战德臣

哈尔滨工业大学　　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

带有条件的运算组合式

$$|x| = \begin{cases} x & if \;\; x > 0 \\ 0 & if \;\; x = 0 \\ -x & if \;\; x < 0 \end{cases}$$

```
(cond  ( <p1>  <e1>)
       ( <p2>  <e2>)
       ...
       ( <pn>  <en>) )
```

```
(define  (abs  x) (cond  ((>  x  0)  x)
                         ((=  x  0)  0)
                         ((<  x  0)  (-  x))))
```

(7)训练一下?

◆问题1：用前缀表示法书写下述表达式

$$\frac{10 + 4 + (8- (12 - (6 + 4 \div 5)))}{3*(6-2)(12-7)}$$

◆问题2：请定义一个过程，求某一数值的立方 a^3

◆问题3：进一步以问题2定义的过程，再定义一个过程，求某两个数值的立方和。a^3+b^3 进一步求 5^3+8^3，并模拟给出计算过程。

战德臣 教授

◆问题4：请定义一个过程，计算下列函数

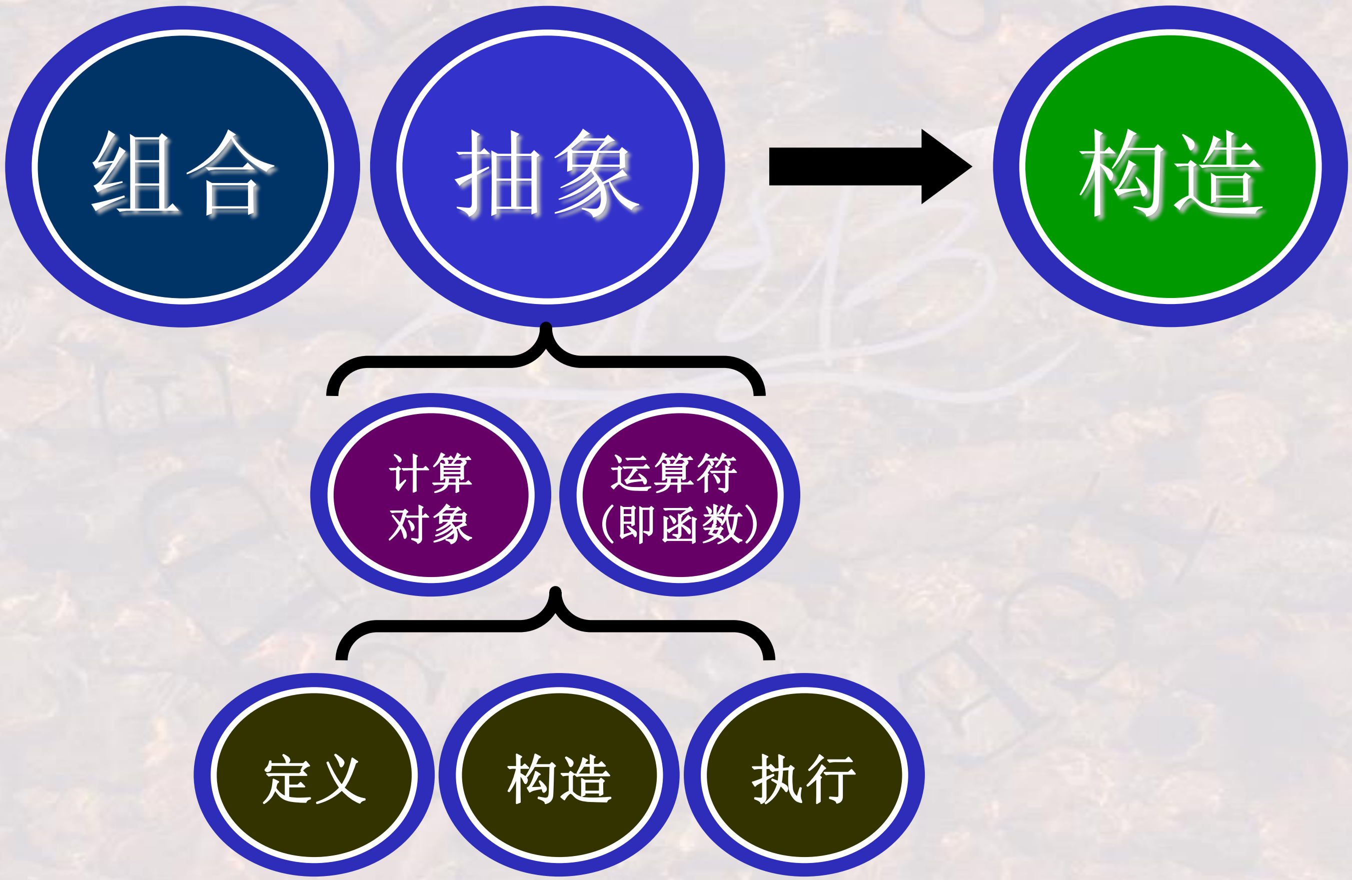
$$f(x)=\begin{cases} x^2-x & \text{if } x>0 \\ 0 & \text{if } x=0 \\ -x^2+x & \text{if } x<0 \end{cases}$$

```
(cond  ( <p1>  <e1>)
       ( <p2>  <e2>)
       ...
       ( <pn>  <en>) )
```

```
(define  (f  x) (cond  ((>  x  0)  (-  (Square x)  x))
                       ((=  x  0)  0)
                       ((<  x  0)  (-  x  (Square x) ))  ))
```

运算式的组合-抽象与构造---程序构造示例
(9)小结

战德臣 教授



递归的概念

战德臣

哈尔滨工业大学　　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

递归(Recursion)

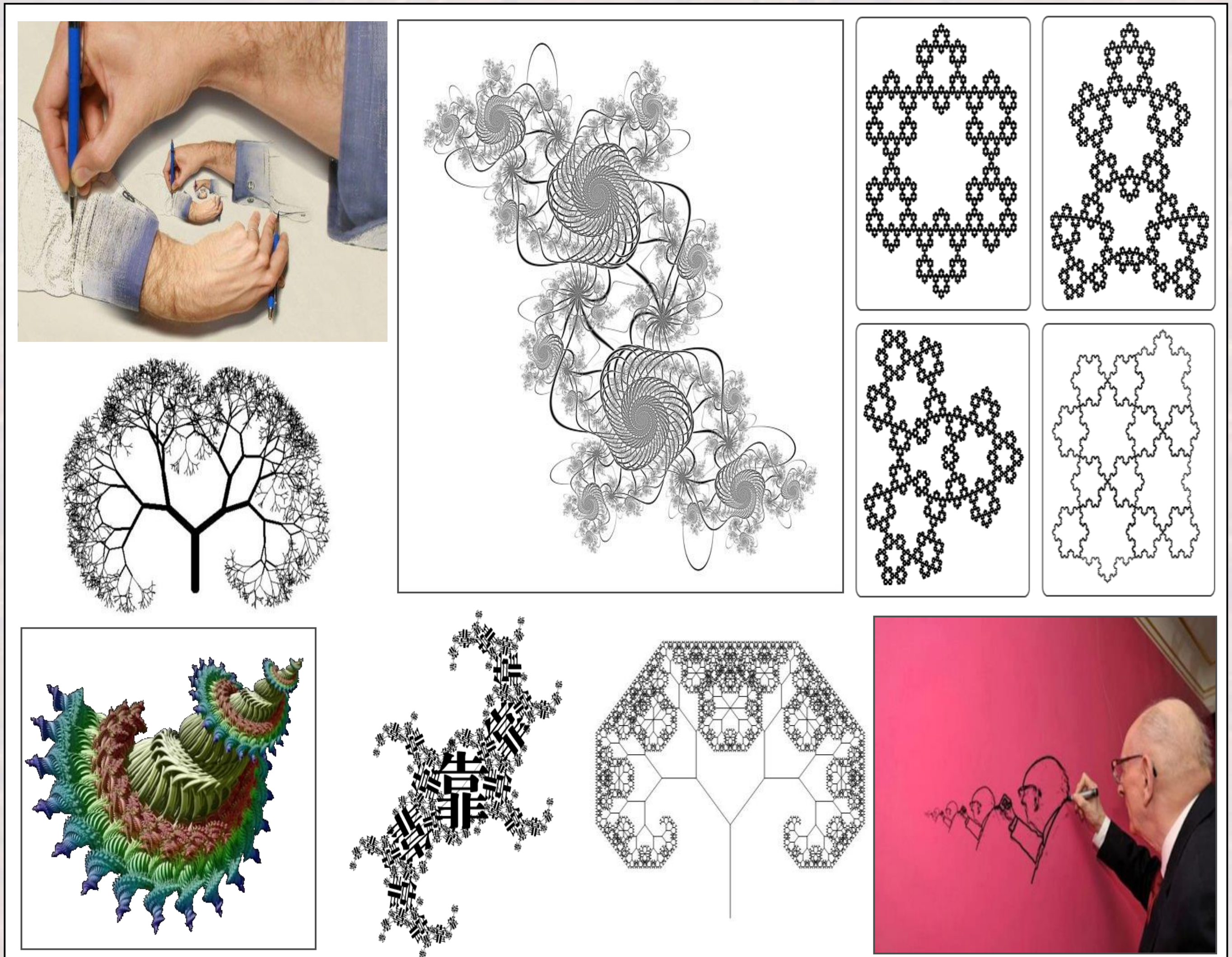
(* •••(* (* (* (* 1 1) 2) 3) 4) ••• n)

怎样在表达中既去掉省略号，而又能表达近乎无限的内容

“从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？(从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？(从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？(从前……)))”

递归的概念
(2)什么情况需要递归?

递归的典型视觉形式---自相似性事物的无限重复性构造



递归的典型视觉形式---自相似性事物的无限重复性构造

数学中的递推式

◆一个数列的第n项a_n与该数列的其他一项或多项之间存在某种对应关系，被表达为一种公式，称为递推式

等差数列递推公式

$a_0=5$

$a_n=a_{n-1}+3$ 当n>=1时

- 第1项(或前K项)的值是已知的—递推基础;
- 由第n项或前n项计算第n+1项—递推规则/递推步骤;
- 由前向后，可依次计算每一项

等差数列的产生

$a_0=5$

$a_1=a_0+3 = 8$

$a_2=a_1+3 = 11$

$a_3=a_2+3 = 14$

$a_4=a_3+3 = 17$

… …

数学中的数学归纳法

◆数学归纳法是一种用于证明与自然数有关的命题正确性的证明方法，该方法能用有限的步骤解决无穷对象的论证问题。

◆由归纳基础和归纳步骤构成：

●假定对一切正整数n, 有一个命题P(n),若以下证明成立, 则P(n)为真。

● (1)归纳基础: 证明P(1)为真;

● (2)归纳步骤: 证明对任意的i, 若P(i)为真，则P(i+1)也为真。

求证命题P(n) “从1开始连续n个奇数之和是n的平方” 即公式: $1+3+5+\cdots+(2n-1)=n^2$成立。

证明:

(1)归纳基础: 当n=1时, 等式成立即1=1;

(2)归纳步骤: 设对任意k, P(k)成立, 即$1+3+5+\cdots+(2K-1)=K^2$.

则 $P(K+1) = 1+3+5+\cdots+(2K-1)+(2(K+1)-1) = K^2+2K+1=(K+1)^2$， 则当P(k)成立时P(K+1)也成立， 根据数学归纳法该命题得证。 证毕。

战德臣 教授

递归是一种表达相似性对象及动作的无限性构造的方法。

■**递归基础**：定义、构造和计算的起点，直接给出；

■**递归步骤**：由前n项或第n项定义第n+1项；由低阶f(k)且k<n，来构造高阶f(n+1)

用递归定义

用递归构造

递归计算/执行

◆递归是一种关于抽象的表达方法**---用递归定义无限的相似事物**

◆递归是一种算法或程序的构造技术**---自身调用自身，高阶调用低阶，构造无限的计算步骤**

◆递归是一种典型的计算/执行过程**---由后向前代入，**直至代入到递归基础，再由递归基础向后计算直至计算出最终结果，即**由前向后计算**

递归的概念

(5)小结

组合 抽象 → 构造 递归

- 用递归定义
- 用递归构造
- 递归计算/执行

递归基础

递归步骤

原始递归函数-复合与递归

战德臣
哈尔滨工业大学　　教授.博士生导师
教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

原始递归函数是接受自然数x或自然数的元组$(x_1,\cdots x_n)$作为参数，并产生自然数的一个映射，记为f(x)或$f(x_1,\cdots x_n)$。接受n个参数的函数称作n元函数。处处有定义的函数被称作全函数，未必处处有定义的函数称作半函数或部分函数。

最基本的原始递归函数，也被称为本原函数有三个：

(1)初始函数：0元函数即常数无需计算；或者常数函数：对于每个自然数n和所有的k, 有$f(x_1,x_2,\cdots,x_K)=n$。

(2)后继函数：1 元后继函数 S，它接受一个参数并返回给出参数的后继数。例如S(1)=2, …, S(x) = x+1, 其中x为任意自然数。

(3)投影函数：对于所有 n≥1 和每个 1≤i≤n 的 i，n 元投影函数 P_i^n，它接受 n 个参数并返回它们中的第 i 个参数，即

$$P_i^n(x_1, x_2, \cdots, x_n) = x_i$$

(1)复合：给定原始递归函数 $f(x_1,\ldots,x_k)$，和 k 个原始递归函数 $g_1,\ldots,g_k$，则f 和 $g_1,\ldots,g_k$的复合是 函数h, 即

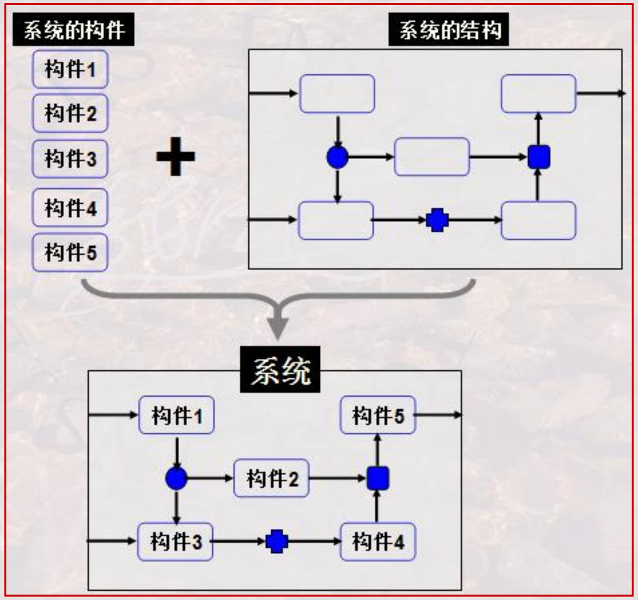
$$h(x_1,\ldots,x_m) = f(g_1(x_1,\ldots,x_m),\ldots,g_k(x_1,\ldots,x_m))$$

简单而言，复合是将一系列函数作为参数代入到另一个函数中，又被称为代入。复合是构造新函数的一种方法。复合是表达组合的一种方法。

$g_1\cdots,g_k$的指令组合关系f vs. 基本指令$g_1,\cdots,g_k$

$g_1\cdots,g_k$的组合关系f vs. 运算组合式$g_1,\cdots,g_k$

结构f vs. 构件$g_1,\cdots,g_k$



(2)原始递归：给定原始递归函数 f 和 g，则新函数h可由 f 和 g递归的定义，其中

$$h(0,x_1,\ldots,x_k) = f(x_1,\ldots,x_k)$$

$$h(S(n), x_1,\ldots,x_k) = g(h(n,x_1,\ldots,x_k),\ n,\ x_1,\ldots,x_k)$$

简单而言，定义新函数h，就是要定义h(0), h(1),…,h(n),…。h(0)直接给出。h(n+1)则由将h(n)和n代入g中来构造。

原始递归是递归地构造新函数的方法，尤其是无限的相似性函数的构造方法。

(* (* …(* (* (* 1 1) 2) 3) … n) S(n))

$g(x_1, x_2) = (*\ \ x_1\ \ S(x_2))$

h(0) = 1

h(S(n)) = g(h(n), n)

h(0) = 1

h(1) = g(h(0), 0) = (* 1 1)

h(2) = g(h(1), 1) = (* (* 1 1) 2)

h(3) = g(h(2), 2) = ((* (* 1 1) 2) 3)

…

h(S(n)) = g(h(n), n)

原始递归函数的构造示例

- 已知：

$f(x)=x$

$g(x_1,x_2,x_3)=x_1+x_2+x_3$, 其中$x,x_1,x_2,x_3$均为自然数

$h(0,x) = f(x)$且 $h(S(n), x) = g(h(n,x),n,x)$

该函数对任一自然数的计算过程为：

$h(0,x)=f(x) =x$

$h(1,x)=h(S(0),x) = g(h(0,x),0,x) = g(f(x),0,x) = f(x) +0+ x =2x$

$h(2,x)=h(S(1),x) = g(h(1,x),1,x)=g(g(f(x),0,x),1,x)=g(2x, 1, x)=3x+1$

$h(3,x)=h(S(2),x) =g(h(2,x),2,x)=g(g(h(1, x),1, x),2, x)= g(g(g(h(0,x),0,x),1,x),2,x)$

$= \cdots = 4x+3$

… …

原始递归函数的构造示例

- 已知：

$f(x)=2$

$g(x_1,x_2,x_3)=x_1$，其中x,x_1,x_2,x_3均为自然数

$h(0,x) = f(x)$且 $h(S(n), x) = g(h(n,x),n,x)$

该函数对任一自然数的计算过程为：

$h(0,x) =f(x) =2$

$h(1,x) =h(S(0),x) = g(h(0,x),0,x) = g(f(x),0,x) = f(x) =2$

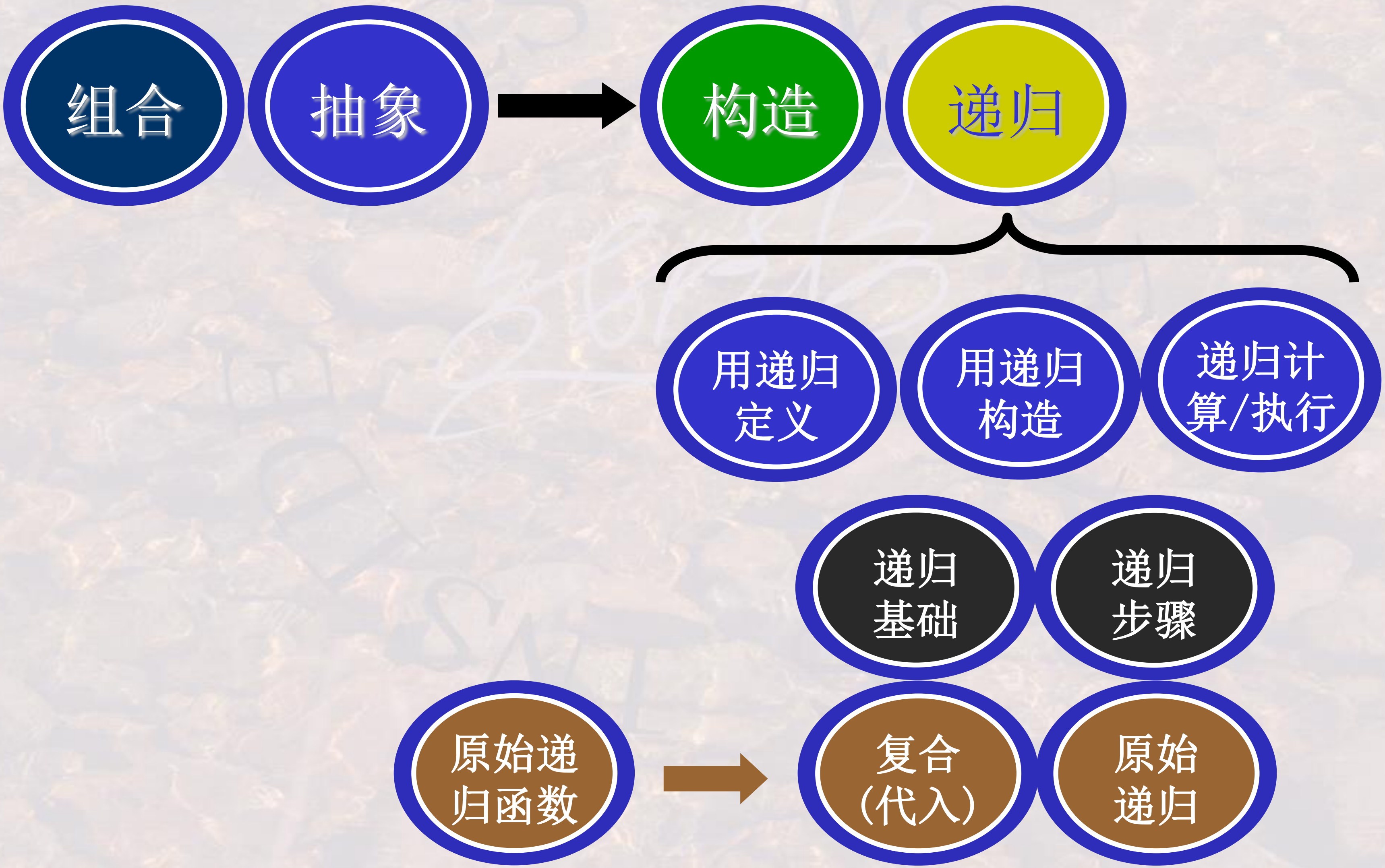
$h(2,x)=h(S(1),x) = g(h(1,x),1,x)=g(g(f(x),0,x),1,x)=g(2, 1, x)=2$

$h(3,x)=h(S(2),x) =g(h(2,x),2,x)=g(g(h(1, x),1, x),2, x) =g(g(g(h(0,x),0,x),1,x),2,x)$

$= \cdots = 2$

… …

原始递归函数-复合与递归
(5)小结



两种不同的递归函数

---递归与迭代

战德臣

哈尔滨工业大学　　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

递归和递推：比较下面两个示例

- Fibonacci数列，无穷数列1，1，2，3，5，8，13，21，34，55，……，称为Fibonacci数列。它可以递归地定义为：

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

递归定义

F(0)=1;

F(1)=1;

F(2)=F(1)+F(0)=2;

F(3)=F(2)+F(1)= 3;

F(4)=F(3)+F(2)= 3+2=5;… …

递推计算/迭代计算/迭代执行

定义是递归的, 但执行可以是递归的也可是迭代的

两种不同的递归函数--递归与迭代
(1)两种不同的递归函数?

递归和递推：比较下面两个示例

- 阿克曼递归函数---双递归函数
- 阿克曼给出了一个不是原始递归的可计算的全函数。表述如下：

$$A(1,0) = 2$$

$$A(0,m) = 1 \qquad m \geq 0$$

$$A(n,0) = n+2 \qquad n \geq 2$$

$$A(n,m) = A(A(n-1,m),m-1) \qquad n,m \geq 1$$

递归定义

函数本身是递归的，
函数的变量也是递归的

m=0时， A(n,0)=n+2；

m=1时， A(n,1)=A**(A(n-1,1),0)=A(n-1,1)+2**， 和A(1,1)=2故A(n,1)=2*n

m=2时， A(n,2)=A**(A(n-1,2),1)=2A(n-1,2)**， 和A(1,2)=A(A(0,2),1)=A(1,1)=2，故A(n,2)= 2^n。

m=3时，类似的可以推出 $\underbrace{2^{2^{2^{N^{2}}}}}_{n}$

递归计算/递归执行

由后向前代入，再由前向后计算

递归和递推：比较下面两个示例

- 阿克曼递归函数---双递归函数---另一种形式

$$A(m,n)=\begin{cases} n+1 & 若m=0 \\ A((m-1),1) & 若n=0 \\ A(m-1,A(m,n-1)) & 若m,n>0 \end{cases}$$

A(1,2) = A(0,A(1,1)) = A(0, A(0,A(1,0))) = A(0, A(0,A(0,1)))=A(0,A(0,2))=A(0,3)=4
。

A(1,3) =A(0, A(1,2))=A(0. <…代入前式计算过程>)=A(0,4)=4+1=5。

…

A(1,n) =A(0, A(1,n-1))=A(0. <…代入前式计算过程>)=A(0,n+1)=n+2。

A(2,1) = A(1，A(2，0)) = A(1，A(1，1)) =A(1，A(0，A(1，0)))
= A(1，A(0，A(0，1))) = A(1，A(0，2)) = A(1，3) =A(0，A(1，2))
= A(0，A(0，A(1，1))) = A(0，A(0，A(0，A(1，0))))
= A(0，A(0，A(0，A(0,1))) = A(0，A(0，A(0，2))) = A(0，A(0，3))
= A(0，4) = 5。

递归和迭代(递推)

◆迭代(递推)：可以自递归基础开始，由前向后依次计算或直接计算；

◆递归：可以自递归基础开始，由前向后依次计算或直接计算；但有些，只能由后向前代入，直到递归基础，寻找一条路径，然后再由前向后计算。

◆递归包含了递推(迭代)，但递推(迭代)不能覆盖递归。

(3)小结?

递归

迭代

运用递归与迭代

战德臣
哈尔滨工业大学　　教授.博士生导师
教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent
Computing for Enterprises & Services,
Harbin Institute of Technology

战德臣 教授

示例：算术表达式的递归定义

首先给出递归基础的定义：

(1)任何一个常数C是一个算术表达式；

(2)任何一个变量V是一个算术表达式；

再给出递归步骤：

(3)如F、G是算术表达式，则下列运算：

F+G，F-G，F*G，F/G是算术表达式；

(4)如F是表达式，则(F)亦是算术表达式。

(5)括号内表达式优先计算，“*”与“/”运算优先于“+”与“-”运算。

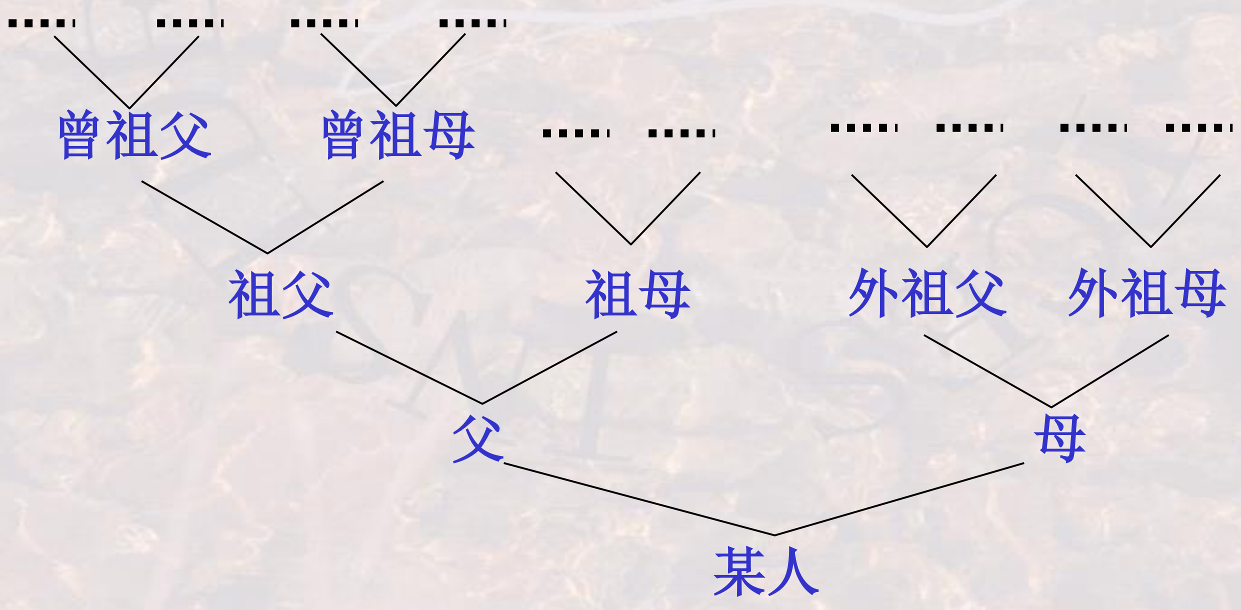
(6)算术表达式仅限于以上形式。

(··· (((100 + (X + Y)) * (Z-Y)) + Z) ···)

示例：“某人祖先”的递归定义

(1)某人的双亲是他的祖先（递归基础）。

(2)某人**祖先**的双亲同样是某人的**祖先**（递归步骤）。



战德臣 教授

示例：简单命题逻辑的形式化递归定义

(1)一个命题是其值为真或假的一个判断语句（递归基础）。

(2)如果X是一个命题，Y也是一个命题，则X and Y， X or Y, Not X也是一个命题。（递归步骤）。

(3)如果X是一个命题，则(X)也是一个命题，括号内的命题运算优先。

(4)命题由以上方式构造。

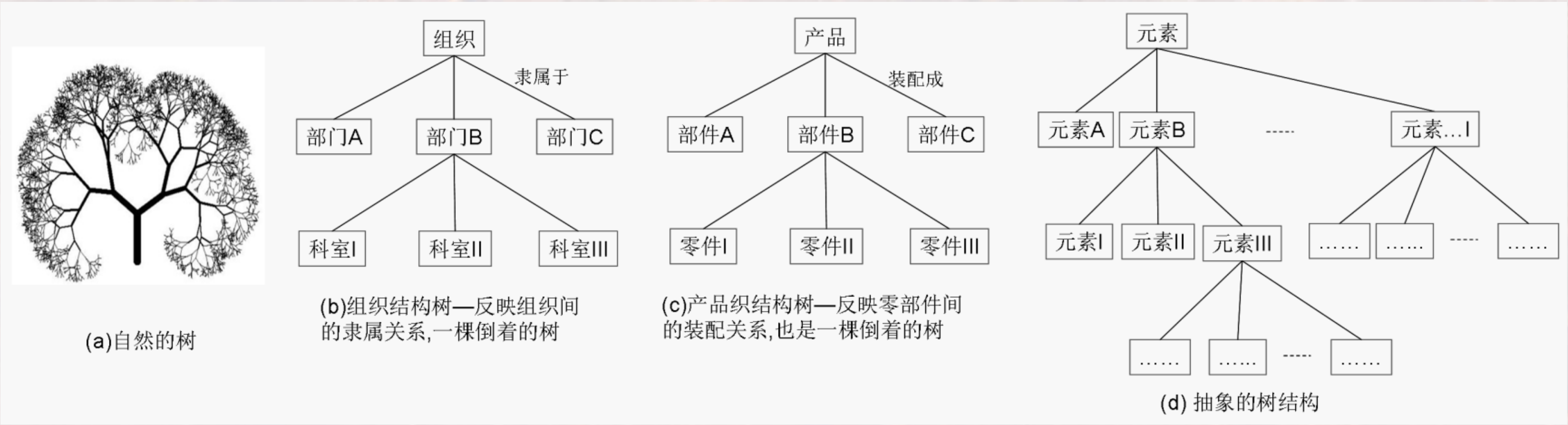
(··· (((M or (X and Y)) and (Y or K)) and Z) ···)

运用递归和迭代

(1)运用递归进行无限自相似性对象的定义?

战德臣 教授

示例：树的形式化递归定义



(a)自然的树

(b)组织结构树—反映组织间的隶属关系,一棵倒着的树

(c)产品织结构树—反映零部件间的装配关系,也是一棵倒着的树

(d) 抽象的树结构

树是包含若干个元素的有穷集合，每个元素称为结点。其中：

(1)有且仅有一个特定的称为根的结点；(递归基础)

(2)除根结点外的其余结点可被分为k个互不相交的集合T_1, T_2,⋯,T_k(k≥0)，其中每一个集合T_i本身也是一棵树，被称其为根的子树。(递归步骤)

运用递归进行程序构造：具有无限的自相似性步骤的表达，自身调用自身，高阶调用递阶

示例：求n!的算法或程序 --*用递归方法构造*

$$n!=\begin{cases}1 & \text{当} n\le 1 \text{时}\\ n\times(n-1)! & \text{当} n>1 \text{时}\end{cases}$$

(define (fact n) (cond ((<= n 1) 1)
((> n 1) (* n fact(n-1)))))

运用递归进行程序构造：具有无限的自相似性步骤的表达，自身调用自身，高阶调用递阶

(define (f n) (cond ((<= n 1) V)

((> n 1) (expression n fact(n-1)))))

(define (expression n m) (* (/ n 2) m)) ——— *这样定义*

$$f(n)=\begin{cases}1 & \text{当}n\le 1\text{时}\\ n/2\times f(n-1) & \text{当}n>1\text{时}\end{cases}$$

(define (expression n m) (+ n m)) ——— *或者这样定义*

$$f(n)=\begin{cases}1 & \text{当}n\le 1\text{时}\\ n+f(n-1) & \text{当}n>1\text{时}\end{cases}$$

运用迭代进行程序构造：具有无限的自相似性步骤的表达，循环-替代-递推

$$n! = \begin{cases} 1 & \text{当} n \le 1 \text{时} \\ n \times (n-1) \times \ldots \times 1 & \text{当} n > 1 \text{时} \end{cases}$$

(*…(* (* (* (* 1 1) 2) 3) 4) …n)

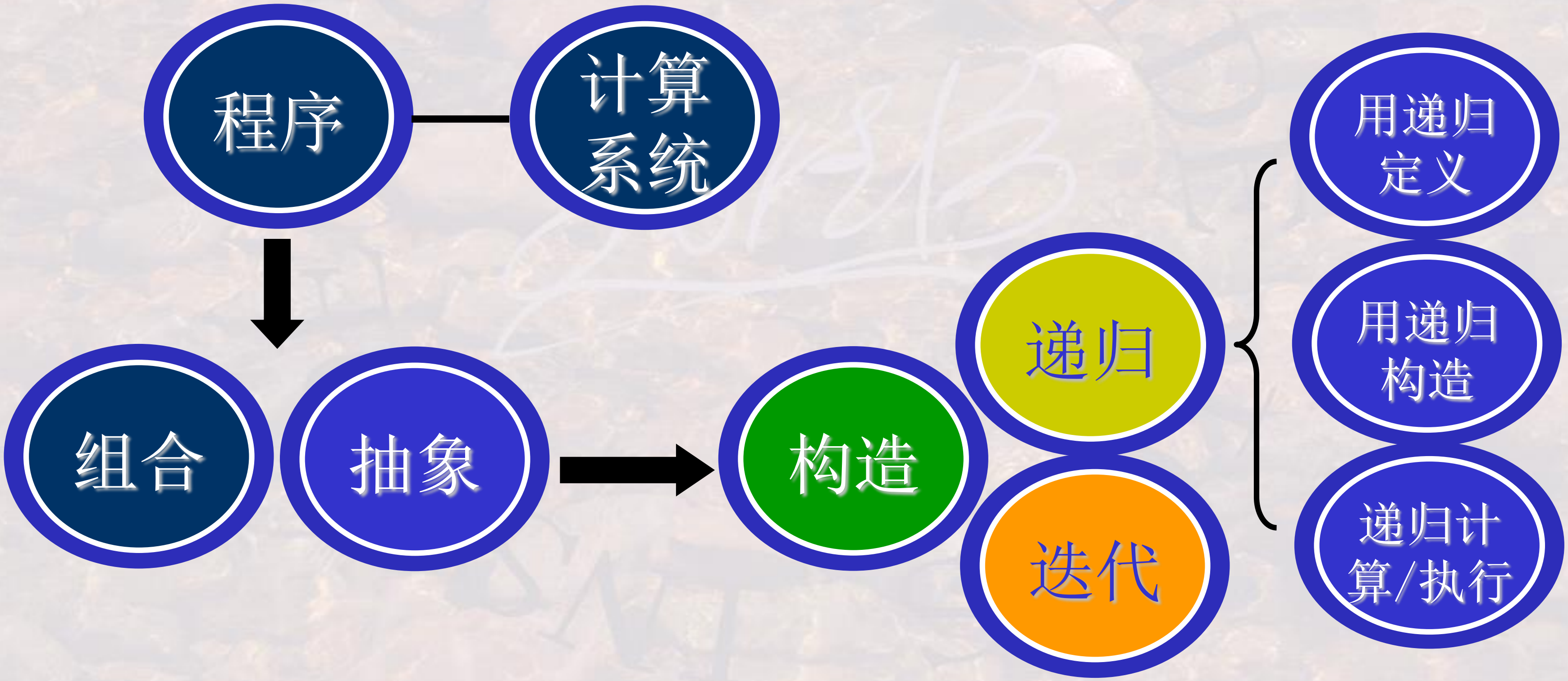
Product ← Product * Counter
Counter ← Counter + 1

```
(define  (fact  n)  (fact-iter  1  1  n))

(define  (fact-iter  product  counter  max-count)
          ( cond  ((>   counter  max-count)  product)
                  ((<=   counter  max-count)
             (fact-iter  (*  counter  product)  (+  counter  1)  max-count ))))
```

运用递归和迭代

(4)小结?



递归与迭代程序的执行

战德臣
哈尔滨工业大学 教授.博士生导师
教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services,
Harbin Institute of Technology

运用递归进行程序构造：具有无限的自相似性步骤的表达，自身调用自身，高阶调用递阶

示例：求n!的算法或程序 --*用递归方法构造*

$$n! = \begin{cases} 1 & \text{当} n \le 1 \text{时} \\ n \times (n-1)! & \text{当} n > 1 \text{时} \end{cases}$$

```
(define  (fact  n) ( cond  ((<=  n  1)  1)
                           ((>  n  1)  (*  n  fact(n-1))) ))
```

递归与迭代程序的执行
(1)运用递归进行程序构造?

战德臣 教授

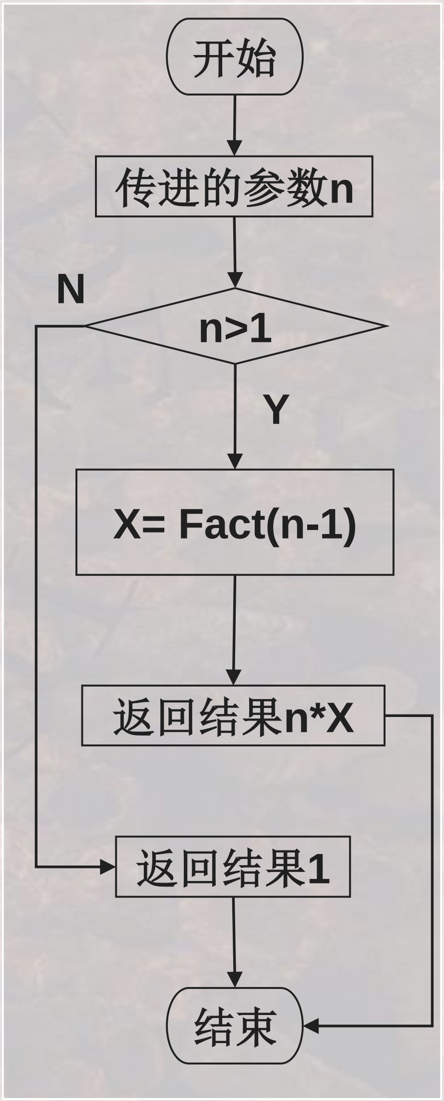
具有无限的自相似性步骤的表达，自身调用自身，高阶调用递阶

示例：求n!的算法或程序 --*用递归方法构造*

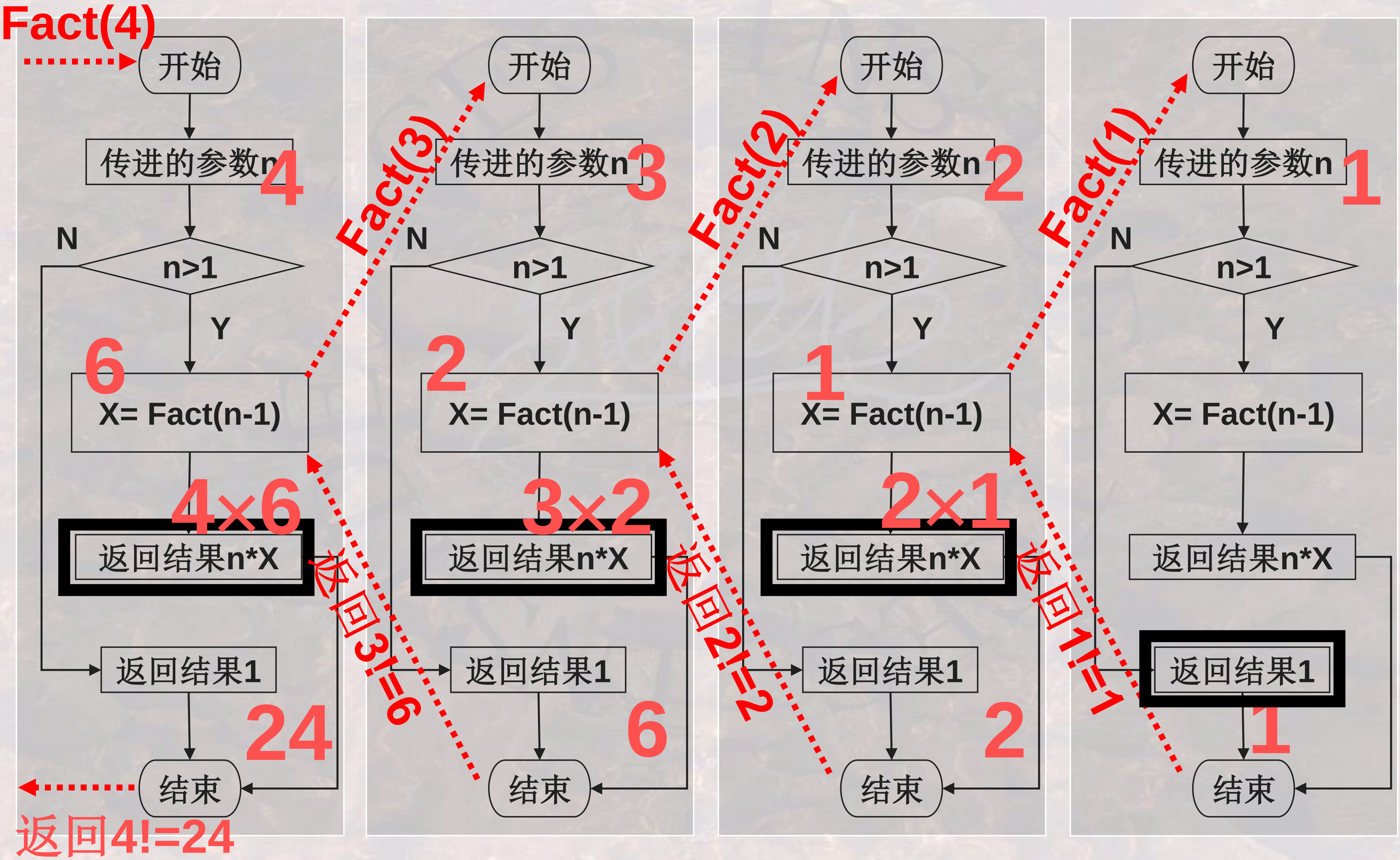
$$n! = \begin{cases} 1 & \text{当} n \le 1 \text{时} \\ n \times (n-1)! & \text{当} n > 1 \text{时} \end{cases}$$

```
long int Fact(int n)
{    long int x;
     If (n > 1)
     { x = Fact(n-1);
   /*递归调用*/
        return n*x;   }
     else return 1;
   /*递归基础*/
}
```

Fact(n)



递归与迭代程序的执行
(2)递归程序的执行过程?



递归与迭代程序的执行
(3)递归程序及其执行过程的另一示例

战德臣 教授

示例：求Fibonacci数列的算法或程序---递归

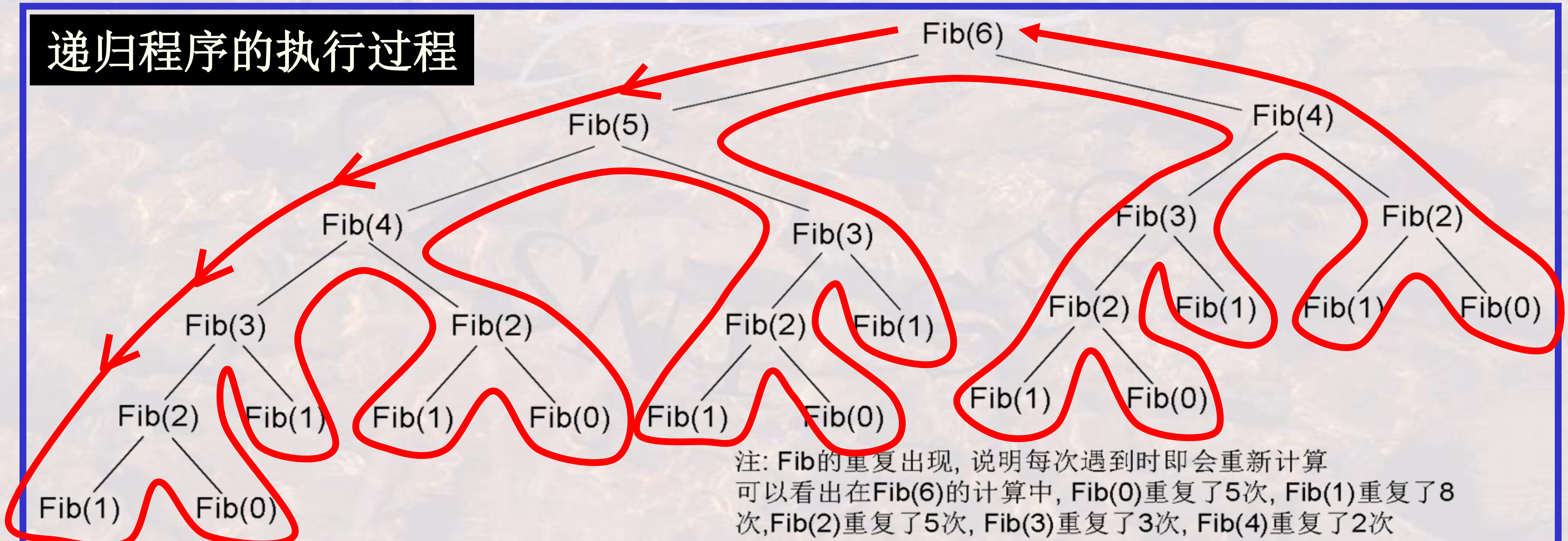
递归定义

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

递归程序

```
(define  (fib  n) ( cond  ((=  n  0)  0)
                          ((=  n  1)  1)
                          ((>  n  1)  (+  (fib  (-  n  1))  (fib  (-  n  2)))) ))
```

递归程序的执行过程



注: Fib的重复出现，说明每次遇到时即会重新计算
可以看出在Fib(6)的计算中，Fib(0)重复了5次，Fib(1)重复了8次,Fib(2)重复了5次，Fib(3)重复了3次，Fib(4)重复了2次

运用迭代进行程序构造：具有无限的自相似性步骤的表达，循环-替代-递推

$$n! = \begin{cases} 1 & \text{当} n \le 1 \text{时} \\ n \times (n-1) \times \ldots \times 1 & \text{当} n > 1 \text{时} \end{cases}$$

(*…(* (* (* (* 1 1) 2) 3) 4) …n)

Product ← Product * Counter
Counter ← Counter + 1

```
(define (fact n) (fact-iter 1 1 n))
(define (fact-iter product counter max-count)
        ( cond ((> counter max-count) product)
               ((<= counter max-count)
          (fact-iter (* counter product) (+ counter 1) max-count ))))
```

(5)迭代程序的执行过程?

运用迭代进行程序构造：具有无限的自相似性步骤的表达，循环-替代-递推

```
(define (fact n) (fact-iter 1 1 n))
(define (fact-iter product counter max-count)
        ( cond ((> counter max-count) product)
               ((<= counter max-count)
  (fact-iter (* counter product) (+ counter 1) max-count ))))
```

(fact 6)

➔ *(fact-iter 1 1 6)*

➔ *(fact-iter (* 1 1) (+ 1 1) 6)* ➔ *(fact-iter 1 2 6)*

➔ *(fact-iter (* 1 2) (+ 2 1) 6)* ➔ *(fact-iter 2 3 6)*

➔ *(fact-iter (* 2 3) (+ 3 1) 6)* ➔ *(fact-iter 6 4 6)*

➔ *(fact-iter (* 6 4) (+ 4 1) 6)* ➔ *(fact-iter 24 5 6)*

➔ *(fact-iter (* 24 5) (+ 5 1) 6)* ➔ *(fact-iter 120 6 6)*

➔ *(fact-iter (* 120 6) (+ 6 1) 6)* ➔ *(fact-iter 720 7 6)*

➔ *720*

战德臣 教授

具有无限的自相似性步骤的表达，循环-替代-递推—迭代

示例：求n!的算法或程序 --*用迭代方法构造*

$$n! = \begin{cases} 1 & \text{当} n \le 1 \text{时} \\ 1 \times 2 \times \ldots (n-1) \times n & \text{当} n > 1 \text{时} \end{cases}$$

```
long int Fact(int n)
{   int  counter;
    long product=1;
    for  counter=1 to n step 1
    { product = product * counter;   }
  /*迭代*/
return product;
}
```

Fact(5)的执行过程

	Counter	Product
初始值		1
循环第1次	1	1
循环第2次	2	2
循环第3次	3	6
循环第4次	4	24
循环第5次	5	120
退出循环	6	120

战德臣 教授

示例：求Fibonacci数列的算法或程序---迭代

递归定义

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

迭代程序

```
(define (fib n) ( fib-iter 1 0 n) )
(define (fib-iter a b count)
        ( cond ((= count 0) b)
              ((> count 0) ( fib-iter (+ a b) a (- count 1) ))))
```

迭代程序的执行过程

a	b	Count		(+ a b)	计算内容
1	0	7		1	初始调用
1	1	6		2	f(0)+f(1)
2	1	5		3	f(1)+f(2)
3	2	4		5	f(2)+f(3)
5	3	3		8	f(3)+f(4)
8	5	2		13	f(4)+f(5)
13	8	1		21	f(5)+f(6)
21	13	0			f(6)+f(7)

关于递归的进一步学习

- 递归是计算技术的典型特征，是以有限的表达方式来表达无限对象实例或无限计算步骤的一种经典的计算思维
- 递归覆盖了重复、迭代和递归，递归是最典型的构造手段
- 递归函数是可计算函数的精确的数学描述---计算理论的重要研究内容;
- (后面将介绍的)图灵机本质上也是递归：图灵可计算函数与递归函数等价，凡可计算的函数都是一般递归函数---丘奇-图灵命题---计算理论的重要研究内容;

(9)小结?

什么是程序?
程序的本质是什么?

递归定义、递归算法、递归计算

程序构造的基本方法：递归与迭代

组合/抽象 → 递归

实例层面：运算组合式
概念层面：计算系统与程序

程序本质---组合、抽象、构造与执行

程序---对基本动作的组合
计算系统---执行程序的系统

计算系统的构造

概念/原理与案例相结合
知识伴随思维，思维贯通知识
相互关联且递进的方式展开与贯通

递归与迭代程序的执行
(9)小结?

战德臣 教授