计算理论

计算机科学与工程学院

序　　言

0. 序言

计算理论是研究计算模型的一门学科.

莱布尼兹（**Leibniz**）的理想：

- 寻求思维的规律，并不是近代人类才开始的努力。两千三百多年前的亚里士多德，这位在科学和哲学上都有伟大贡献的历史巨人，最先总结了人类思维形式一些规则。
- 从亚里士多德开始，人们发现思维形式有一些规则是十分严格的，它们和思维的具体内容没有直接关系。很多推理并不依赖所论证的具体内容。如：

  假如下雨地就会湿。今天下雨了，故今天地湿。

  人都有一死。苏格拉底是人，所以苏格拉底也会死。

  抽象化后，$A \rightarrow B$，$A \vdash B$。　　(假言推理）

  又如，　$(A \vee B) \wedge \neg B \vdash A$　　（析取三段论）

- 把推理规则变成演算规则，莱布尼兹首先创立了“逻辑演算”。再进一步，就可以设法造出能进行这种演算的机器，即所谓逻辑演算机。莱布尼兹曾幻想过，一个那样的黄金时代即将来临：两个哲学家发生哲学分歧时，用不着辩论，只要把笔拿在手里，在计算工具面前坐下来，两个人面对面笑嘻嘻地说：“让我们来计算一下吧！”谁对谁错，看计算的结果就知道了。

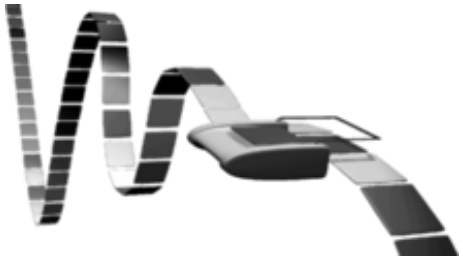
- 由此可见，在莱布尼兹看来，“思维”不过是一种复杂的“计算”而已。

- 什么是“计算”？

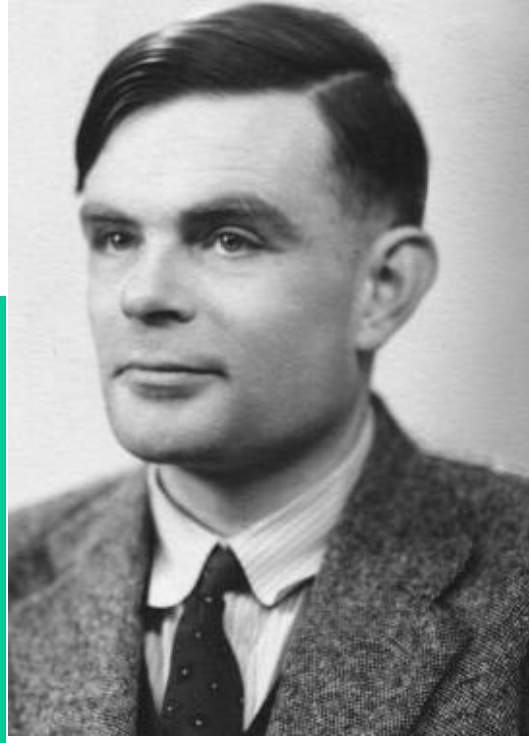
计算理论在1936年取得了重大的突破。英国学者图灵(A.M.Turing）发表了《论可计算的数及其对判定问题的应用》。他严格地用数学方法定义了什么是“计算”（能左移、右移、读、写的磁带机的一次运行），即著名的“图灵机”。还证明了，存在一个图灵机，它能执行任何计算过程，（只要给它所要求计算过程的程序）——通用计算机是存在的。这在理论上排除了巴贝奇的继承者们去构造通用计算机的疑虑，同时也为如何构造提供了理论指导。（今天计算机界没有Nobel奖，但有Turing奖）。就计算能力而言，今天的所有计算机与图灵机等价。

Church-Turing论题:一切算法都等价于Turing机 。

（TM例）



计算理论的开拓者—图灵



阿兰·图灵（Alan Turing），1912年6月23日出生于英国伦敦。其祖父曾获得剑桥大学数学荣誉学位，图灵的父亲J·M·图灵早年就读于牛津大学历史系，图灵的母亲E·S·斯托尼(Stoney) 曾就读于巴黎大学文理学院，图灵是他们的次子。

1931年-1934年，图灵在英国剑桥大学国王学院（King’s College）学习。1936年，主要研究可计算理论，并提出“图灵机”的构想。1936年-1938年，主要在美国普林斯顿大学做博士研究与丘奇一同工作，博士论文题目为“以序数为基础的逻辑系统”，在数理逻辑研究中产生了深远的影响。1938年，返回剑桥从事研究工作。1951年，年仅39岁的图灵，被选为英国皇家学会会员。1954年，图灵逝世，床头放着一个被咬了一口的苹果。

* 图灵机的例子：y=x+2

Writing $s_0 = B$, $s_1 = 1$ consider the Turing machine with alphabet {1}:

q_1	B	R	q_2
q_2	1	R	q_2
q_2	B	1	q_3
q_3	1	R	q_3
q_3	B	1	q_1.

We can check the computation:

$$\begin{array}{cccccccccc} B111, & B111, & \ldots, & B111B, & B1111, & B1111B, & B11111 \\ \uparrow & \uparrow & & \uparrow & \uparrow & \uparrow & \uparrow \\ q_1 & q_2 & & q_2 & q_3 & q_3 & q_1 \end{array}$$

一. 本课的性质以及研究的内容

- 任何一门学科都有它的基础和它的基本问题，如物质的本质是什么？有机体生命的基础和起源是什么？
- 什么是计算机科学的基础？什么是计算机科学的基本问题？

  诸如什么是计算？哪些问题是可计算的？哪些是不可计算的？什么是可判定的？什么是不可判定的？
- 这些问题就是计算理论要讨论的问题。

- 计算理论课程是计算机专业区别于其他专业的主要课程之一.

- **停机问题:判断任意一个程序是否会在有限的步骤之内结束运行的程序是否存在?**
- 图灵回答了该问题：**不存在这样的程序**。
- **证明（反证法）：假设存在这样的程序**，即：存在程序H(P,I)可以给出程序P在输入I时是否停机的判断：若P在输入I时可停机，H输出“停机”，反之输出“不停机（死循环）”（注： 停机 与 死循环 是对立的），即可导出下面的矛盾：
- 依据假设，可以设计一个调用程序H的主程序K如下：首先，它调用H(P,P)，如果H(P,P)输出“死循环”，则K(P)停机，反之K(P)死循环。即K(P)做与H(P,P)的输出相反的动作。 K(P)的程序如下：
- int H(P,I); // 这里的H函数有两种返回值：死循环 或 停机
  int K(P) {
  if (H(P,P) == 死循环) { //程序本身可以被视作数据
  return 停机; //主程序K停机 }
  else { // H(P,P) == 停机
  while(1){} // 这里K会死循环 }
  }
- 视K程序为参数P，观察K(**K**)的运行：若H(K , K)输出死循环，实际观察到K(K)停机，但由H的定义知二者矛盾。反之，H(K, K)输出停机，K(K)实际死循环，两者仍然矛盾。
- 因此，假设程序H存在引发不可避免的矛盾，表明开始的假设不成立，结论是：**不存在这样的程序。**证明结束。
- 又称：“H是不可计算函数”、“停机问题不可解”。
- 以停机问题为基础，发现了大量不可计算问题，如“铺砖问题”等。
  LINK

这个问题，计算机永远无法解决

2017-01-17 点这里 更精彩 → 书圈

热文导读 | 点击标题阅读

老师们的“花式点名”方法盘点，点名方式哪家强？

计算机可以驾驶汽车，可以控制火星探测器登

性质：该课是计算机科学的理论课。

- 计算理论：就是研究理论计算机的科学。
- 理论计算机：是研究计算机的理论模型，研究计算机的本质，也就是把计算机看成一个数学系统。(因为计算机科学的基本思想和模型在本质上是数学——离散的。)

内容：

- **形式语言与自动机理论：**

  正规文法与有限自动机 (正规语言) 、
  上下文无关文法与下推自动机(上下文无关语言)
  图灵机(递归可枚举语言)

- **可计算性理论：** 什么是可计算？
- **计算复杂性理论：** 时间复杂性 、 空间复杂性。
- **递归函数** (简单介绍)

二. 学习目的：

- **了解这些计算理论**

我们知道计算机不论从它的诞生还是它的快速发展过程都没有离开计算理论，也就是它是在计算理论指导下诞生和发展的。本课所涉及的都是计算机科学的基本问题。不首先了解它们，是很难理解计算机科学的。作为计算机科学与技术专业的本科生和研究生应该了解这些计算理论。

- **培养能力**

此课可以培养学生抽象逻辑思维和形式化思维的能力。**Harry Lewis**语: 计算理论是我们这个领域的“共同潜意识”。现在软件企业**CTO**都看好抽象思维好的学生。

- **为学习《编译原理》做准备**

- **教学参考书:** (1)蒋宗礼,姜守旭编.《形式语言与自动机理论》（第2版）.清华大学出版社.2007; **(2)Harry R.Lewis** 等人著. 张立昂、刘田译.《计算理论基础》(第2版). 北京：清华大学出版社. **2006** ; (3) M.Davis著《可计算性-复杂性-语言》,张立昂等译.清华出版社,**1989.**
- 教学要求: **40**学时, **2**次书面作业和编程作业,，期中期末考核。

第一章

形式语言概述

语言是人们交流思想的工具。按照语言的形成，可将语言分成两类：自然语言和人工语言（形式语言）。

一. 自然语言

如汉语、英语、法语、日语等等都是自然语言。

形成：是大多数人经过长期地社会实践逐渐形成的。

特点：种类繁多，内容丰富，表达能力强。

缺点：具有地方性，不便互相交流。有时不够精确，有多义性。比如汉语中的“打”字，具有多种解释。如打伞、打扑克、打醋、打人、一打袜子等等。因此自然语言不适合计算机的程序设计语言。

二.形式语言

字母表上字符串的集合。如计算机的各种程序设计语言、数理逻辑中的谓词演算语言等都属于形式语言。

形成：是少数人经过严格地形式定义确定的语言。

特点：定义准确，无歧义性。

在五十年代Chomsky建立了形式语言的理论体系，从此它发展很快，形式语言的研究已成为计算机科学的一个重要领域。

形式语言：按一定规律构成的句子(或符号串)的有限或无限的集合。特点是：1、高度的抽象化(采用数学公式-来描述语言的结构关系)；2、是一套演绎系统(形式语言本身的目的就是要用有限的规则来推导语言中无限的句子,提出形式语言的哲学基础也是想用演绎的方法来研究自然语言)。例如：$\{a^n b^n \mid n \geq 0\}$

描述形式语言有两种方法：

- 生成法
- 识别法。

生成法：用文法给出产生该语言的所有句子的规则。根据这些规则可以产生语言中每个句子。这些规则就叫生成式或产生式。

例如，下边是个描述“十进制数”的文法：

G=({F,I,D,N}, {.,0,1,2,3,4,5,6,7,8,9}, P, F)

令**F**——“十进制数”、　　**I**——“无符号整数”、

D——“十进制小数”、　**N**——“数字”

于是该文法的产生式集合**P**中产生式如下：

F→I|D|ID

I→N|NI

D→.I

N→0|1|2|3|4|5|6|7|8|9

例如利用此文法产生**3.14:**

F⇒ID⇒ND

⇒N.I⇒3.I

⇒3.NI⇒3.1I

⇒3.1N⇒3.14

识别法：核心是一个自动机。对于给定的符号串可以由自动机识别出是否为给定语言中合法的句子。

自动机的具体的例子以后再介绍。

1-1 形式语言基本概念

形式语言必须规定所用基本符号集合，这就是字母表。

一.字母表

字母表：符号的有限集合。通常用V或者∑表示。

例如 V＝{a,b,c} 。

二. 符号串

符号串：是由字母表中的符号组成的序列。

例如，aabbcc就是上述字母表V上的一个符号串。

符号串的长度：即是符号串所含符号个数。

例如符号串α＝aabbcc 用|α|表示α的长度，则 |α|＝6。

空符号串：不含任何符号的符号串，通常用ε表示。

显然|ε|＝0 。

三.符号串的“连接”运算“◦”

例符号串x=abc，y=cba，x与y的连接构成符号串z, 则

$z=x\circ y=abc\circ cba=abccba$

显然连接运算“◦”满足可结合性且有幺元ε，即对任何符号串x,y,z有

$(x\circ y)\circ z=x\circ(y\circ z)$　　$x\circ\varepsilon=\varepsilon\circ x=x$

对符号串的连接可以写成乘幂的形式，即对任何符号串x有：

$x\circ x=x^2$

$x\circ x\circ x=x^3$

一般地

$x^{n-1}\circ x=x^n$

$x^m\circ x^n=x^{m+n}$

$(x^m)^n=x^{mn}$

四．符号串集合的乘积

令A和B是符号串的集合，A与B的乘积记作AB，且

$$AB=\{x\circ y \mid x\in A\wedge y\in B\}$$

例如，A＝{a,b,ab}，B={0,1}， 则

AB＝{a0,b0,ab0,a1,b1,ab1}

由于符号串集合的乘积的运算是可结合的，所以也可写成乘幂的形式。即A是符号串集合，则

$AA=A^2$

$A^mA^n=A^{m+n}$

$(A^m)^n=A^{mn}$

当两个集合中有一个集合是空集时，则 它们的乘积为空集。即ΦA=AΦ=Φ。

五．字母表的闭包V^+与V^*

令V是个字母表。则

V——由V中符号构成的长度为1的符号串的集合。

V^2——由V中符号构成的长度为2的符号串的集合。

V^3——由V中符号构成的长度为3的符号串的集合。

于是

V^k＝{w|w是由V中的符号构成的符号串，且|w|=k }

V^0={ε}

$V^+=V\cup V^2\cup V^3\cup V^4\cup\ldots$

$V^*=V^0\cup V\cup V^2\cup V^3\cup V^4\cup\ldots$

V^*是由V中符号构成的任意长度的符号串(所有符号串)构成的集合。

例如，V＝{0,1}

V^+={0,1,00,01,10,11,000,001,010,011,100,101,110,111,…}

V^*={ε,0,1,00,01,10,11,000,001,010,011,100,101,110,111,…}

六．语言

定义：设V是个字母表, $L \subseteq V^*$,则称L是V上的一个语言。

例如，V＝{0,1}

$L_1=\Phi$

L_2={0, 00, 000，0000，00000，……}

L_3={1, 11, 111, 1111, 11111,……}

上述 L_1、L_2、L_3 都是V上的语言。

七.句子

设L是V上的语言，如果w∈ L，则称w是 L 中的一个句子。

例如，000就是L_2中的一个句子。

1.2 文法概念

文法是语言生成器中的最重要的一类，为了定义语言，
文法不仅作为一个“装置”，给出语言的句子的结构，而
且本身也是一个数学系统。
例如：前边定义“十进制数”的文法。

G=({F,I,D,N}, {.,0,1,2,3,4,5,6,7,8,9}, P, F)

F—十进制数、　　I—无符号整数、

D—十进制小数、N—数字

于是该文法的产生式集合P中产生式如下：

F→I|D|ID　　I→N|NI　　D→.I

N→0|1|2|3|4|5|6|7|8|9

可见一个文法中有两种符号。

终极符

非终极符

1.文法（Grammar）定义

一个文法G是个有序四元组，记作

$G=(V_N, V_T, P, S)$，其中

V_N—非终极符(变元)集合，用大写英文字母表示。

V_T—终极符集合。

这里$V_N \cap V_T=\Phi$。有时记作 $V_N \cup V_T = V$

P—生成式(也叫产生式)的集合。

产生式的形式: $\alpha\rightarrow\beta$，其中 $\alpha\in V^+$，$\beta\in V^*$

$\alpha\rightarrow\beta$的含义是：可以用符号串β代替符号串α。

另外如果有$\alpha\rightarrow\beta$，$\alpha\rightarrow\gamma$，$\alpha\rightarrow\delta$

可简记成$\alpha\rightarrow\beta \mid \gamma \mid \delta$

S—开始变元，$S\in V_N$。

【例1-2.1】下面是定义只含有＋和*运算的算术表达式的文法。

$G_1=(V_N,V_T,P,E)$

$V_N=\{E,T,F\}$，

$V_T=\{a,b,+,*,(,)\}$

$P=\{E\rightarrow E+T,\quad E\rightarrow T,$

$T\rightarrow T*F,\quad T\rightarrow F,$

$F\rightarrow(E),\quad F\rightarrow a,\quad F\rightarrow b\}$

如：(a+a*b)即可由E推导出来.

【例1-2.2】

$G_2=(\{S\},\{0,1\},P,S)$

$P=\{S\rightarrow 0S1|01\}$

文法中使用的符号通常作如下约定：

（1）用大写英文字母表示变元。S通常表示开始变元。

（2）用小写的a,b,c,…表示终极符。

（3）用x,y,z,…表示终极符串，即$x,y,z,\ldots \in V_T^*$。

（4）用α,β,γ,…希腊字母表示既含有终极符，也含有非终极符的符号串，即$\alpha,\beta,\gamma,\ldots \in (V_N \cup V_T)^*$。

2.句型（Sentential form）

设文法 $G=(V_N, V_T, P, S)$，则

（1）S是个句型。

（2）若αβγ是个句型,且β→δ是P中的一个产生式，则αδγ也是一个句型。

按此定义，对于文法G_2来说， P＝{S→0S1|01}

S，0S1，00S11，000111都是句型。

3.句型的推导（派生）

设文法$G=(V_N, V_T, P, S)$，若$\alpha\beta\gamma$是G的一个句型，且$\beta \to \delta \in P$，则$\alpha\delta\gamma$也是一个句型，我们就称为可由句型$\alpha\beta\gamma$直接推导出$\alpha\delta\gamma$,记作$\alpha\beta\gamma \Rightarrow_G \alpha\delta\gamma$。

如果没有其它文法，不会产生混淆的情况下，此推导可简记成$\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$。

符号“$\Rightarrow$”表示句型之间的直接推导(派生)关系。

如果有$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \ldots \Rightarrow \alpha_n$，则表示由$\alpha_1$可以间接推导出$\alpha_n$，可以写成$\alpha_1 \Rightarrow^* \alpha_n$，

符号“$\Rightarrow^*$”表示句型之间经过多步间接推导的关系。

符号“$\Rightarrow^k$”表示句型之间经过k步间接推导的关系。

4. 文法产生的语言

设文法 $G=(V_N, V_T, P, S)$，令集合

$L(G)=\{w \mid w \in V_T^* \text{且} S \Rightarrow^* w\}$

则称L(G)是由G产生的语言。

其中每个$w \in L(G)$是文法G产生的句子。

在例1-2.2中，$P=\{S \rightarrow 0S1 \mid 01\}$

有 $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111$,

所以$L(G_2)=\{0^n1^n \mid n \geq 1\}$

【例1-2-X】 G_3=({S,B,C,H},{a,b,c}，P，S)

P: (1) S→aSBC (2) S→aBC (3) CB→HB

(4) HB →HC (5) HC →BC

(6) aB→ab (7) bB→bb (8) bC→bc

(9) cC→cc

求L(G_3)。

解. $S \Rightarrow^* a^{n-1}S(BC)^{n-1}$ (产生式(1)使用n-1次)

$\Rightarrow a^n(BC)^n$ (产生式(2)使用一次)

(详细见超连接)

$\Rightarrow^* a^n b^n c^n$ (产生式(9)使用多次)

所以 $L(G_3)=\{a^n b^n c^n \mid n\geq 1\}$

【例1-2.4】 $G_4=(\{S,A,B\},\{a,b\}, P, S)$

P: $S\rightarrow aB|bA$, $A\rightarrow a|aS|bAA$, $B\rightarrow b|bS|aBB$

求证 $L(G_4)$中的每个句子里的a和b的个数相同。

证明：令$N_a(w)$表示w中a的个数，

1).先证 即(任一$w\in L(G_4)$,则$N_a(w)= N_b(w)$)

先用归纳法(对G中推导$S\Rightarrow^{*}\alpha$的步数n作归纳)证明如

下结论：

如果 $S\Rightarrow^{*}\alpha$ ， 则 $N_{a+A}(\alpha)=N_{b+B}(\alpha)$。

其中$N_{a+A}(\alpha)$表示 α 中a和A的个数总和。

(1) 当n=1时，此推导一定是用产生式 S→aB|bA，于是有 S⇒aB 或 S⇒bA，结论成立。

(2) 假设n≤k 时，结论成立。即如果有$S \Rightarrow^{n} \alpha_1$ (表示从S经n步推出α_1)， 则 $N_{a+A}(\alpha_1)= N_{b+B}(\alpha_1)$ 。

(3) 当 n=k+1 时，不妨设 $S \Rightarrow^{k} \alpha_1 \Rightarrow \alpha_2$，则由(2)得

$N_{a+A}(\alpha_1)= N_{b+B}(\alpha_1)$

下面讨论推导$\alpha_1 \Rightarrow \alpha_2$，由于$\alpha_1$中的变元只有三种，所以分三种情况讨论：

① 此派生是对α_1中的变元S作代换，必用产生式 S→aB|bA，显然不论使用哪一个产生式，都能得出结论 $N_{a+A}(\alpha_2)= N_{b+B}(\alpha_2)$。

② 此派生是对α_1中的变元A作代换，必用产生式 A→a|aS|bAA，显然不论使用哪一个产生式，都能得出结论$N_{a+A}(\alpha_2)= N_{b+B}(\alpha_2)$。

③ 此派生是对α_1中的变元B作代换，必用产生式B→b|bS|aBB，显然不论使用哪一个产生式，都能得出结论：$N_{a+A}(\alpha_2)= N_{b+B}(\alpha_2)$。

综上所述，上述命题成立。

任取$w\in L(G_4)$，于是有推导$S \Rightarrow^* \alpha \Rightarrow w$,由上述结论得$N_{a+A}(\alpha)= N_{b+B}(\alpha)$

而在最后一步推导$\alpha \Rightarrow w$中，要消去α中的变元，必用产生式A→a或B→b,显然仍有$N_{a+A}(w)= N_{b+B}(w)$,此时w中A和B的个数都为0，于是有$N_a(w)= N_b(w)$.所以$w\in L$，于是有 $L(G_4) \subseteq L$。(至此,本命题结论已得证.更深入的可考虑(2))

2) 再证 $L \subseteq L(G_4)$

任取$w\in L$,显然$N_a(w)= N_b(w)$,令$N_a(w)=n$,对n作归纳，证出$w\in L(G_4)$。

(1) n=1 时，则w=ab,或w=ba，在G_4中有推导：

S⇒aB⇒ab或S⇒bA⇒ba，所以有 $w\in L(G_4)$

(2) 假设n≤k时，结论成立。即w∈L, $N_a(w)=N_b(w)$，$N_a(w)\le k$，则有w∈L(G_4)， 即G_4中有推导$S\Rightarrow^{*}w$ 。

(3) 当n=k+1时，(即$N_a(w)= k+1$,)，因为w中的最左符号不是a就是b。

下面分两种情况讨论。

a) w的最左符号是a ,不妨设$w=a^ibx$,$(i\ge1)$, $x\in\{a,b\}^{+}$,

如果i=1，w=abx , 则$N_a(x)=N_b(x)$,且 $N_a(x)=k$,由假设(2)得 x∈L(G_4)， 所以 $S\Rightarrow^{*}x$,于是有推导：

$S\Rightarrow aB\Rightarrow abS\Rightarrow^{*}abx=w$ ， 所以w∈L(G_4)。

如果i>1, $w=a^ibx$，可将w写成$w=a^ibw_1bw_2\ldots bw_i$，其中 $N_a(w_j)=N_b(w_j)$ $(1\le j\le i)$, 这些w_j中，可能$w_j=\varepsilon$或$w_j\in L$。

如果$w_j\in L$，又$N_a(w_j)\le k$，由归纳假设得$w_j\in L(G_4)$，

即$S\Rightarrow^{*}w_j$ 于是在G_4中有推导：

$S\Rightarrow aB\Rightarrow aaBB\Rightarrow^{*}a^iB^i$, 再往下推导。

$S \Rightarrow aB \Rightarrow aaBB \Rightarrow^* a^iB^i$ ，再往下推导。

如果$w_j = \varepsilon$ ， 则对应的第j个B可用产生式B→b替换，可得$B \Rightarrow bw_j$。

如果 $w_j \neq \varepsilon$,则对应得第j 个B可用产生式B→bS替换，

又因为$S \Rightarrow^* w_j$,可得$B \Rightarrow^* bw_j$, 最后得推导：

$$S \Rightarrow aB \Rightarrow aaBB \Rightarrow^* a^iB^i \Rightarrow^* a^ibw_1B^{i-1} \Rightarrow^* a^ibw_1bw_2B^{i-2} \Rightarrow^* \ldots \Rightarrow^*$$

$$a^ibw_1\,bw_2bw_3\ldots bw_i=w$$

即有 $S \Rightarrow^* w$, $w \in L(G_4)$ 。

b) 当w的最左符号是b时，不妨设$w=b^iax\ (i\geq 1)$, $x \in \{a,b\}^+$，类似可证。

于是，对于n=k+1 时，命题成立。

即，如果$w \in L$，则$w \in L(G_4)$, 所以 $L \subseteq L(G_4)$。

最后得，$L=L(G_4)=\{w \mid w \in \{a, b\}^+ \text{且} N_a(w) = N_b(w)\}$。

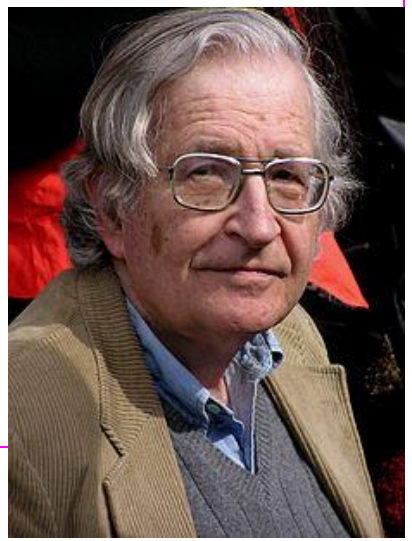
1-3 文法的分类

按照文法的产生式的结构不同，将文法分成四类，称之为**Chomsky**分类。

分别称之为**0**型、**1**型、**2**型、**3**型。

令文法 $G=(V_N, V_T, P, S)$ $V_N \cup V_T = V$，

具体的结构形式如下表所示：



Chomsky is an American linguist, cognitive scientist. He is a professor of MIT, the “father of modern linguistics”.

类型	文法结构		产生式形式	限制条件
0	短语结构文法 Phrase Structure		$\alpha\rightarrow\beta$	$\alpha\in V^{+},\beta\in V^{*}$ e.g.
1	上下文有关文法 Context Sensitive （CSG）		$\alpha_1A\alpha_2\rightarrow\alpha_1\beta\alpha_2$	$\lvert\alpha_1\beta\alpha_2\rvert\geq\lvert\alpha_1A\alpha_2\rvert$ α_1，$\alpha_2\in V^{*}$ $A\in V_N$，$\beta\in V^{+}$ e.g.
2	上下文无关文法 Context Free （CFG）		$A\rightarrow\alpha$	$A\in V_N,\alpha\in V^{+}$ e.g.
3	正规文法	右线性文法	$A\rightarrow xB$，$C\rightarrow y$	$A,B,C\in V_N$ $x,y\in V_T^{+}$ e.g.
		左线性文法	$A\rightarrow Bx$，$C\rightarrow y$	

按照此定义，判定上一节给定的四个文法是何类型：

$G_1=(V_N,V_T,P,E)$

$V_N=\{E,T,F\}$，　$V_T=\{a,b,+,*,(,)\}$

$P=\{E\to E+T,\ E\to T,\ T\to T*F,\ T\to F,$

$F\to(E),\quad F\to a,\ F\to b\}$

$G_2=(\{S\},\{0,1\},P,S)$　$P=\{S\to 0S1|01\}$

$G_3=(\{S,B,C,H\},\{a,b,c\},P,S)$

P: (1) $S\to aSBC$　(2) $S\to aBC$　(3) $CB\to HB$

(4) $HB\to HC$　(5) $HC\to BC$

(6) $aB\to ab$　(7) $bB\to bb$　(8) $bC\to bc$

(9) $cC\to cc$

$G_4=(\{S,A,B\},\{a,b\},P,S)$

P: $S\to aB|bA$, $A\to a|aS|bAA$, $B\to b|bS|aBB$

G_1、G_2、G_4是2型文法，即上下文无关文法，

而文法G_3是1型文法。

四种文法所产生的语言，分别叫作

0型语言——递归可枚举集(r.e)

(recursively enumerable set)

1型语言——上下文有关语言(CSL)

(Context Sensitive Language)

2型语言——上下文无关语言(CFL)

(Context Free Language)

3型语言——正规集 (regular set)

可以看出，各类文法之间有向上兼容性，即

3型语言⊆2型语言⊆ 1型语言⊆ 0型语言。

文法与自动机之间的对应关系：

识别这些语言的自动机分别是：

0型语言 ——————图灵机

1型语言(CSL)————线性界限自动机

2型语言(CFL)————下推自动机

3型语言(正规集)——有限自动机

有限自动机、下推自动机以及图灵机的内容后边将详细介绍。

本章重点：掌握文法概念和类型。

- **作业题：**

1．给定文法G=({S,A,B,C,D,E},{0,1},P,S)，其中P：

S→ABC，AB→0AD，AB→1AE，AB→ε，D0→0D，D1→1D，E0→0E，E1→1E，C→ε，DC→B0C，EC→B1C，0B→B0，1B→B1

试写出句子01100110的派生过程。

2．设计下列各文法G，使得它们分别是：

（1）G是个上下文无关文法，且

$L(G)=\{a^i b^j c^k \mid i,j,k\geq 1\}$。

（2）G是个正规文法，且

$L(G)=\{a^i b^j c^k \mid i,j,k\geq 1\}$。

（3）G是个上下文无关文法，且

$L(G)=\{ ww^R \mid w\in\{0,1\}^+ \}$。其中$w^R$是w的逆转，例如w=001, 则$w^R$=100.

第一阶段，**词法分析**。词法分析的任务是：输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个的**单词**（亦称**单词符号**或简称**符号**），如基本字（begin、end、if、for、while 等）、标识符、常数、算符和界符（标点符号、左右括号等等）。例如，对于 FORTRAN 的循环语句

DO 150 I = 1，100

词法分析的结果是识别出如下的单词符号：

基本字	DO
标　号	150
标识符	I
等　号	=
整常数	1
逗　点	,
整常数	100

这些单词是组成上述 FORTRAN 语句的基本符号。单词符号是语言的基本组成成份，是人们理解和编写程序的基本要素。识别和理解这些要素无疑也是翻译的基础。如同将英文翻成中文的情形一样，如果你对英语单词不理解或对构词法不熟悉，那就谈不上进行正确的翻译。在词法分析这阶段的工作中所依循的是语言的**构词规则**。

back

*例:L_3= {a^i, i≥1 }，其语法是S→ aS|a。

第二阶段，**语法分析**。语法分析的任务是：在词法分析的基础上，根据语言的**语法规则**（文法规则），把单词符号串分解成各类**语法单位**（语法范畴），如“**短语**”、“**子句**”、“**句子**”（“**语句**”）、“**程序段**”和“**程序**”。通过语法分解，确定整个输入串是否构成一个语法上正确的“**程序**”。语法分析所依循的是语言的语法规则。例如，在一般的面向科学、工程计算的语言中，符号串

X + 0.618乘 Y

代表一个“**算术表达式**”。因而，语法分析的任务就是识别这个符号串属于“**算术表达式**”这个范畴。

* 例:L_2={ a^ib^i, i≥1 } ,其文法是S→aSb|ab

back

【例1-2.3】 G_3=({S,B,C},{a,b,c}，P，S)

P: (1) S→aSBC (2) S→aBC (3) CB→HB

(4) HB →HC (5) HC →BC

(6) aB→ab (7) bB→bb (8) bC→bc (9) cC→cc

所以 $L(G_3)=\{a^n b^n c^n \mid n\geq 1\}$。

在编译程序中，到“代码生成”阶段（含语义分析）不再使用文法产生式，而是利用元语言来解释每个语法规则的语义：为每个规则附加一个语义子程序，来完成这个语句要做的动作。这种方法叫做：“语法制导的翻译方法”。

(1) $E \to E^{(1)} + E^{(2)}$ {E·VAL := $E^{(1)}$·VAL + $E^{(2)}$·VAL}

(2) E → 0 {E·VAL := 0}

(3) E → 1 {E·VAL := 1}

第一个产生式的语义动作规定了：E的语义值E·VAL 等于 $E^{(1)}$ 和 $E^{(2)}$ 的语义值 $E^{(1)}$.

back

*例:L_0= {0^i, | i是2的非负整数次幂 }。

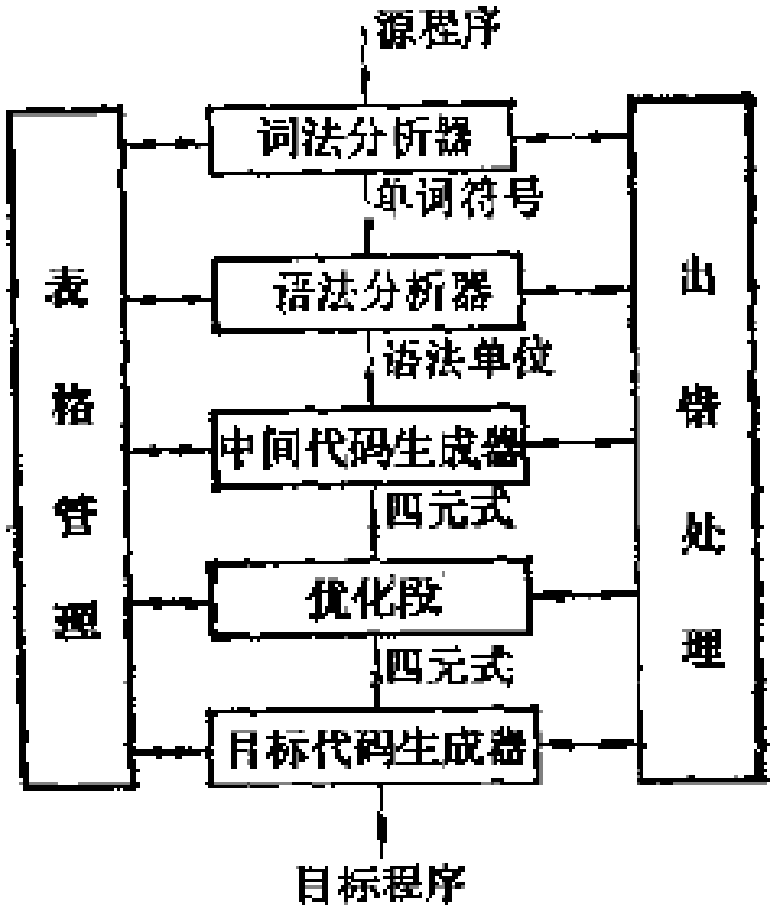


图0.1 编译程序总框

G_0: S→ACOB

CO→OOC

CB→DB

OD→DOO

CB→E

AD→AC

AC→F

FO→OF

OE→EO

AE→ε

FB→ε

back

Almost any language one can think of is context-sensitive; the only known proofs that certain languages are not CSL's are ultimately based on diagonalization. These include L_u of Chapter 8 and the languages to which we may reduce L_u, for example, the languages proved undecidable in Chapter 8. We shall prove in Section 9.4 that there are recursive languages that are non-CSL's, and in Chapter 12 we shall refine this statement somewhat. In both cases the proofs proceed by diagonalization.

The universal language

Define L_u, the "universal language," to be $\{\langle M, w\rangle \mid M \text{ accepts } w\}$. We call L_u "universal" since the question of whether any particular string w in $(\mathbf{0} + \mathbf{1})^*$ is accepted by any particular Turing machine M is equivalent to the question of whether $\langle M', w\rangle$ is in L_u, where M' is the TM with tape alphabet $\{0, 1, B\}$ equivalent to M constructed as in Theorem 7.10.

Example 8.1 Let $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$ have moves:

$$\delta(q_1, 1) = (q_3, 0, R),$$

$$\delta(q_3, 0) = (q_1, 1, R),$$

$$\delta(q_3, 1) = (q_2, 0, R),$$

$$\delta(q_3, B) = (q_3, 1, L).$$

Thus one string denoted by $\langle M, 1011 \rangle$ is

$$111010010001010011000101010010011$$

$$000100100101001100010001000100101111011$$

Note that many different strings are also codes for the pair $\langle M, 1011 \rangle$, and any of these may be referred to by the notation $\langle M, 1011 \rangle$.

Examples

- This grammar generates the canonical non-context-free language $\{a^n b^n c^n \mid n \geq 1\}$:

1. $S \to aSBC$
2. $S \to aBC$
3. $CB \to HB$
4. $HB \to HC$
5. $HC \to BC$
6. $aB \to ab$
7. $bB \to bb$
8. $bC \to bc$
9. $cC \to cc$

The generation chain for aaa bbb ccc is:

$$
\begin{aligned}
&S \\
&\Rightarrow_1 aSBC \\
&\Rightarrow_1 a\boldsymbol{aSBC}BC \\
&\Rightarrow_2 aa\boldsymbol{aBC}BCBC \\
&\Rightarrow_3 aaaB\boldsymbol{HB}CBC \\
&\Rightarrow_4 aaaB\boldsymbol{HC}CBC \\
&\Rightarrow_5 aaaB\boldsymbol{BC}CBC \\
&\Rightarrow_3 aaaBBC\boldsymbol{HB}C \\
&\Rightarrow_4 aaaBBC\boldsymbol{HC}C \\
&\Rightarrow_5 aaaBBC\boldsymbol{BC}C \\
&\Rightarrow_3 aaaBB\boldsymbol{HB}CC \\
&\Rightarrow_4 aaaBB\boldsymbol{HC}CC \\
&\Rightarrow_5 aaaBB\boldsymbol{BC}CC \\
&\Rightarrow_6 aa\boldsymbol{ab}BBCCC \\
&\Rightarrow_7 aaa\boldsymbol{bb}BCCC \\
&\Rightarrow_7 aaab\boldsymbol{bb}CCC \\
&\Rightarrow_8 aaabb\boldsymbol{bc}CC \\
&\Rightarrow_9 aaabbb\boldsymbol{cc}C \\
&\Rightarrow_9 aaabbbc\boldsymbol{cc}
\end{aligned}
$$

More complicated grammars can be used to parse $\{a^n b^n c^n d^n \mid n \geq 1\}$, and other languages with even more letters.

back

一阶逻辑中的自然推理系统**G**：

(p) $\alpha_1, \cdots, \alpha_n \vdash \alpha_i (i = 1 \cdots n)$ 假设公理

4. 推理规则

($\neg$) $\dfrac{\Gamma, \neg\alpha \vdash \beta, \Gamma, \neg\alpha \vdash \neg\beta}{\Gamma \vdash \alpha}$ 非联结词规则

($\wedge_+$) $\dfrac{\Gamma \vdash \alpha \text{ 且 } \Gamma \vdash \beta}{\Gamma \vdash \alpha \wedge \beta}$ 合取词引入

($\wedge_-$) $\dfrac{\Gamma \vdash \alpha \wedge \beta}{\Gamma \vdash \alpha}, \dfrac{\Gamma \vdash \alpha \wedge \beta}{\Gamma \vdash \beta}$ 合取词消去

($\vee_+$) $\dfrac{\Gamma \vdash \alpha}{\Gamma \vdash \alpha \vee \beta}, \dfrac{\Gamma \vdash \beta}{\Gamma \vdash \alpha \vee \beta}$ 析取词引入

($\vee_-$) $\dfrac{\Gamma, \alpha \vdash r \text{ 且 } \Gamma, \beta \vdash r \text{ 且 } \Gamma \vdash \alpha \vee \beta}{\Gamma \vdash r}$ 析取词消去

($\rightarrow_+$) $\dfrac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \alpha \rightarrow \beta}$ 蕴涵词引入

($\rightarrow_-$) $\dfrac{\Gamma \vdash \alpha \text{ 且 } \Gamma \vdash \alpha \rightarrow \beta}{\Gamma \vdash \beta}$ 蕴涵词消去

($\leftrightarrow_+$) $\dfrac{\Gamma \vdash \alpha \rightarrow \beta \text{ 且 } \Gamma \vdash \beta \rightarrow \alpha}{\Gamma \vdash \alpha \leftrightarrow \beta}$ 等价词引入

($\leftrightarrow_-$) $\dfrac{\Gamma \vdash \alpha \leftrightarrow \beta}{\Gamma \vdash \alpha \rightarrow \beta}, \dfrac{\Gamma \vdash \alpha \leftrightarrow \beta}{\Gamma \vdash \beta \rightarrow \alpha}$ 等价词消去

命题逻辑公式的定义

(1)单个命题变量A、B、C…是公式;

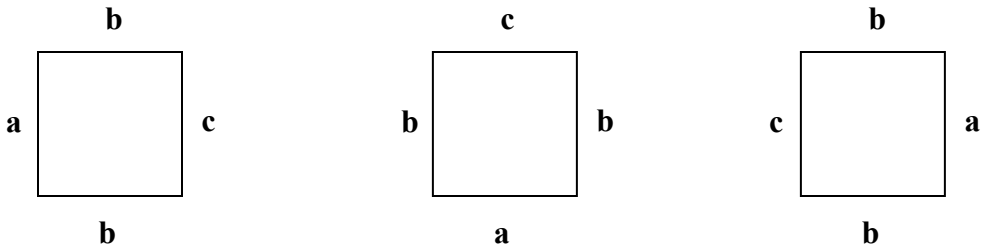
(2)如果P、Q是公式，则 ¬P、P∧Q、P∨Q、P→Q、 P↔Q也是公式;

(3)只有有限次地应用(1)～(2) 才是公式。

BACK

原始的 **Domino Problem** 定义如下：

给定了一个由方型骨牌组成的有穷集合，这些骨牌大小相同(比如，全都具有单位面积)而且不同的方块的边上以不同的方式着了色，铺砖时相邻的砖边缘颜色必须相同，再假定每一骨牌（骨牌型）都有无穷多个 **COPY**．不允许旋转或翻转一骨牌，这样，例如下面的三个骨牌是不同的类型：



一骨牌集称为(在无穷平面上)可解的，当且仅当使用骨牌集中的骨牌能铺盖整个平面。

（BACK）