计算机语言的发展：机器语言、汇编语言、高级语言、命令语言。

翻译程序—编译和解释（笔译与口译/整篇提交与单句提交）[解释方式易于查错，执行效率低]区别在于是否生成目标代码

程序语言一般分为低级语言和高级语言两大类

面向机器的语言指的是由 0/1 代码组成的语言，其特点是计算机可直接识别，但可读性差，理解困难。在此基础上产生了与人类自然语言比较接近的高级语言。

翻译程序能够将源程序转换成与其等价的目标程序。

对编译程序而言，输入数据是源程序，输出数据是目标程序。

如果编译生成的目标程序是机器代码，则源程序的执行分两大阶段（编译）和（执行）；如果目标程序是汇编语言程序，则源程序的执行分三大阶段（编译），（汇编）和（执行）。

- 中序遍历：中缀表示
- 前序遍历：波兰表示/前缀表示
- 后序遍历：逆波兰表示/后缀表示

中缀表示

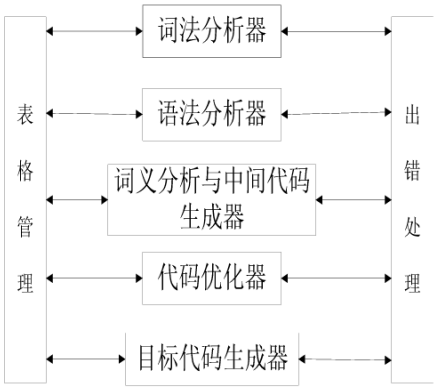
(a+①b)*(c+②d)+③e/f

波兰表示——也就是前缀表示

+③*+①a b+②c d/ef

逆波兰表示——也就是后缀表示

a b +①c d +②*ef/+ ③



(x+6)/y-z*p+m　　波兰表示+-/+x6y*zpm　逆波兰表示 x6+y/zp*-m+

中间代码的特点：简单规范、与机器无关、易于优化与转换

对代码进行等价变换以求提高执行效率，提高运行速度和节省存储空间[与机器无关的优化-与机器有关的优化]

与机器无关的优化：局部优化[常量合并、公共字表达式的提取]、循环优化[强度削减、代码外提]

while(i<10){T1=4*I;i=i+1;}==while(i<10){T1=T1+4}

与机器有关的优化：寄存器的利用、体系结构、存储策略、任务划分

目标代码的形式

具有绝对地址的机器指令

汇编语言形式的目标程序

模块结构的机器指令

源程序-前段　目标程序-编译后端

程序设计语言的语言处理程序是一种系统软件

- 编译过程中，语法分析器的任务是（ ）。
  - ☐ B. 分析单词串是如何构成语句和说明的
  - ☐ C. 分析语句和说明是如何构成程序的
  - ☐ D. 分析程序的结构
- 编译程序与具体的机器有关，与具体的语言无关。

高级语言的分类：

面向过程的语言（非结构化/结构化[顺序、选择、循环]/程序的层次性和抽象性不高）

面向对象的语言（以对象为核心，具有封装性、多态性和继承性）

面向应用的语言

专用语言

要在某一机器上为某种语言构造一个编译程序，必须掌握三方面的内容：源语言、目标语言、编译方法

Chomsky—文法　克林—自动机

- 文法
  - 阐明词法和语法的一种工具
  - 形式化语言理论的基本概念
  - 以有穷的集合刻画无穷的集合

> 文法
> 为一个四元组:G =（VT，VN，P，S）
> VT：终结符(Terminal)集
> VN：非终结符集，VT∩VN=Φ
> S：开始符号(Start Symbol)，S∈VN
> P：产生式集合

方幂：$x^0=\varepsilon$; $x^1=x$; $x^2=xx$; ……; $x^n=x^{n-1}x$

{a，b，c，d}* = {ε ，a，b，c，d，aa，ab，ac，ad，ba，bb，bc，bd，…，aaa，aab，aac，aad，aba，abb，abc，…}

标识符的文法：G =(V_T，V_N，P，S) V_T={a, b, c, 1, 2, 3} V_N={N, L,D}

P:N →L|NL|ND　L →a|b|c　D →1|2|3

N 是开始符号

简单算术表达式的文法：G =({id，+，-，*，/，(，)}，{E}，P，E)

P：E → E+E|E-E|E*E|E/E|(E)|id

二进制整数的文法：B →0|1　B →B0|B1　S →B

正整数的文法：N →1|2|3|4|5|6|7|8|9　　N →N0|N1|…|N9

能被 5 整除的文法：N →1|2|3|4|5|6|7|8|9　N →N0|N1|…|N9　S →N0|N5|0|5

十进制正实数：D →N.T　D →0|0.T|N.0　N →1|2|3|4|5|6|7|8|9　N →N0|N1|…|N9

T →1|2|3|4|5|6|7|8|9　T →0T|1T|…|9T

设 G[S]是一文法，如果符号串 x 是从识别符号推导出来的，即 S ⇒* x，则称 x 是该文法的句型。句型是一种抽象。若 x 仅由终结符号串组成，即 S ⇒* x，且 x∈V_T* ，则称 x 为该文法的句子。文法 G 所产生的语言定义为集合{x|S ⇒* x ，其中 S 为文法识别符号，且 x∈V_T*}。可用 L(G)表示该集合。文法描述的语言是该文法一切句子的集合。

L(G)={0^n1^n|n>=1}　S→0S1　S→01

L= {0^m1^n|m,n≥ 1}　S→0S|0A　A→1A|1

文法 G[S]:S → AB　A → aA|ε　B →bBc|bc　描述的语言 L(G)={$a^mb^nc^n$|m>=0,n>=1}

L(G)={$a^nb^mc^md^n$|n>=0,m>=1}　S→A|aSd　A→bAc|bc

{x|x 是长度为偶数的 0、1 串}　S→00S|01S|10S|11S|ε

描述语言 L={a^mb^n|n≥ m ≥ 1}的文法为：A→Ab|aAb|ε

最左推导：每次推导都施加在句型的最左边的语法变量上

不确定的自顶向下句型分析：出现回溯现象、效率低、编程复杂

已知文法 G(S):

S →(R)|a|^　R →T　T →S,T|S

写出句型 k=(a,(T),(S,T))的短语,直接短语和句柄

短语:

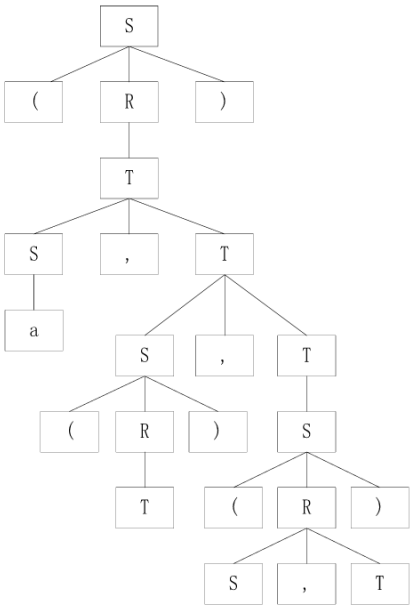
a　T　(T)　S,T　(S,T)　(T),(S,T)

a,(T), (S,T)　　(a,(T), (S,T))

直接短语:

a　T　S,T

句柄:a



文法类型：0 型文法、1 型文法/上下文有关文法[产生式 a→β 满足 |a|≤|β|]、2 型文法/上下文无关文法[|a|≤|β|，产生式 a→β 中的 a 必须是变元]、3 型文法/正规文法[右线性文法 A→aB 或 A→a、左线性文法 A→Ba 或 A→a]

文法的二义性：**E→E+E|E*E|(E)|id**

对于句子 **id+id*id**，有如下两个最左推导：**E⇒E+E ⇒id+E ⇒id+E*E ⇒id+id*E ⇒id+id*id**

E ⇒E*E ⇒E+E*E ⇒id+E*E ⇒id+id*E ⇒id+id*id

E→-EE|-E|a|b|c --a-bc 是二义性文法

设有文法 G[S]：S→dAB A→aA|a B→Bb|ε ，G[S]产生的语言是 L(G)={da^nb^m|n>=1,m>=0}，等价的正规文法为：S→aA A→aA|aB|a B→bB|ε

S→abcA A→bcB B→a 等价的正规文法是：S→aA A→bB B→cC C→bD D→cE E→a

单词一般可以分为 5 类：关键字、标识符、常数、运算符、界限符

正规式	正规集
0|1	{0, 1}
(0|1)(0|1)	{00,01,10,11}
01	{01}
0*	{ε,0,00,000,...}
(0|1)*	{ε, ? ,...}

a|b=b|a
a|(b|c)=(a|b)|c
(ab)c=a(bc)
a(b|c)=ab|ac
(a|b)c=ac|bc
ε a=a
a ε =a

正规文法转换为正规式：A → xB B → y A = xy

A → xA A → y A = x*y

A → x A → y A = x|y

正规式转换为正规文法：R=a(a|d)* S →aA A →aA A →dA A →ε

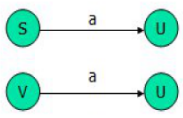
(0|1)*11 (0|1)* S→A|0S|1S A→1B B→1E E→0E|1E|ε

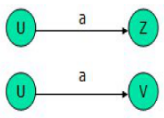
(0|1)*000 S→A|0S|1S A→0B B→0E E→0

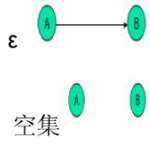
A→[B B→X]|BA X→Xa|Xb|a|b 对应的正规表达式：$[(a|b)^+]^+$

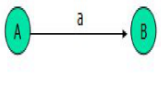
对于左线性正规文法 对于右线性正规文法

U→a U→Va U→a U→aV



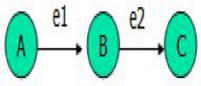


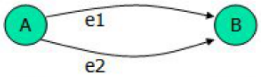


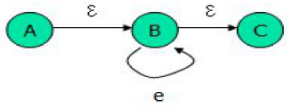


a

e1e2 e1|e2 e*







确定的有穷自动机 DFA：当前状态和下一个输入字母唯一地确定了下一个状态

不确定的有穷自动机 NFA

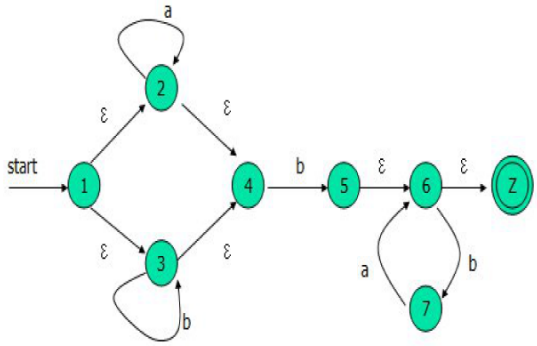
NFA 可以有若干个初始状态，DFA 只有一个；NFA 的状态转换函数不是单值，DFA 相反。DFA 是 NFA 的特例。

构造以下正规式的 NFA：

R=(a*|b*) R=(ab)*

构造与正规表达式 R=(a*|b*)b(ba)*等价的 DFA

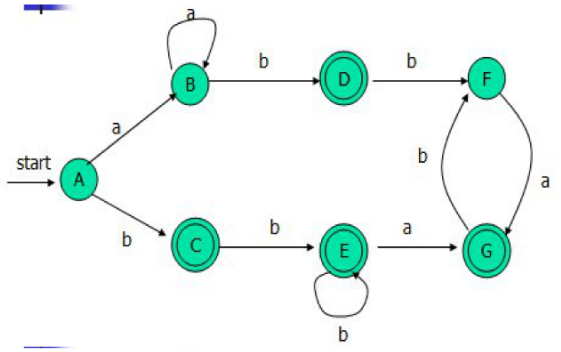
1 构造 NFA



2 转换过程

I	I_a	I_b
A[1,2,3,4]	B[2,4]	C[3,4,5,6,Z]
B[2,4]	B[2,4]	D[5,6,Z]
C[3,4,5,6,Z]	--	E[3,4,5,6,7,Z]
D[5,6,Z]	--	F[7]
E[3,4,5,6,7,Z]	G[6,Z]	E[3,4,5,6,7,Z]
F[7]	G[6,Z]	
G[6,Z]	--	F[7]

3 构造 DFA



一个上下文无关文法是 LL1 文法的充分必要条件是：

- 对每个非终结符 A 的不同产生式，A→α ， A→β 满足
  - FIRST(A→α)∩ FIRST (A→β)=O
- 对每个非终结符 A 的不同产生式， A→α ， A→ ε 满足
  - FIRST(A→α)∩ FOLLOW (A)=O

若有文法 G[S]：S→aAld A→bAS|ε 证明这是一个 LL1 文法

FIRST(S→aA)={a} FIRST (S→d)={d} FIRST(S→aA)∩ FIRST (S→d)= O

FIRST(A→ bAS)={b} FOLLOW(A)=FIRST(S)={a,d,# } FIRST(A→ bAS)∩ FOLLOW(A)=O

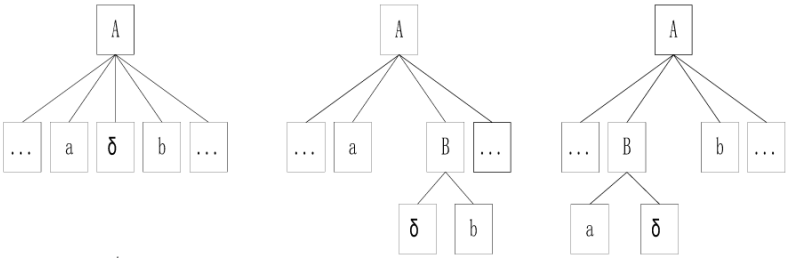
LL1 文法的转换：提取左公因子

- A → α β 1| α β 2|…| α β n
- A → α (β 1| β 2|…| β n) A → α A’ A’= β 1| β 2|…| β n

  消除左递归[直接左递归 A→Aβ 间接左递归 A → Bβ B → Aα]
- S → Sa S → b
- S → bS’ S’ →aS’| ε

将文法 G[S] S→ SalNb|c N → Sd|Nelf 改为无左递归的文法：第二个式子转换为 N→NbS’dlcS’dlNelf

可改写为：S→NbS’lcS’ S’ →aS’lε N→fN’lcS’dN’ N’ →bS’dN’leN’lε



对任意两个终结符最多只有三种关系中的一种成立，则称之为算符优先文法

FIRSTVT(B)={b|B ⇒+b…或者 B ⇒+Cb…}

LASTVT(B)={a|B ⇒+ … a 或者 B ⇒+ … aC}

- 计算 FIRSTVT(A)的方法
  - 若有产生式 A →a…或 A →Ba…，则 a∈ FIRSTVT(A)
  - 若 a∈ FIRSTVT(B)，且有产生式 A →B…，则有 a∈ FIRSTVT(A)
- 计算 LASTVT (A)的方法
  - 若有产生式 A → … a 或 A →… aB ，则 a∈ LASTVT(A)
  - 若 a∈ LASTVT(B)，且有产生式 A →… B ，则有 a∈ LASTVT(A)

FOR 每条产生式 P →X1X2X3…Xn DO

FOR i:1 TO n-1 DO

BEGIN

①IF Xi 和 Xi+1 均为终结符号，THEN 置 Xi=Xi+1

② IF i<=n-2 且 Xi 和 Xi+2 均为终结符号，但 Xi+1 为非终结符号 ，THEN 置 Xi=Xi+2

③ IF Xi 为终结符号，而 Xi+1 为非终结符号

THEN FOR FIRSTVT(Xi+1) 中的每个元素 a DO 置 Xi<a

④IF Xi 为非终结符号，而 Xi+1 为终结符号

THEN FOR LASTVT(Xi) 中的每个元素 a DO 置 a>Xi+1

END

已知文法 G[S]: S →aSb|P P →bPc|bQc Q →Qala 该文法是否是算符优先文法？请证明。

算符优先关系矩阵

	a	b	c
a	<,>	<,=	>
b	<	<,>	=
c		>	>

自下而上语法分析方法的基本思想是：从待输入的符号串开始，利用文法的规则步步向上进行规约，试图规约到文法的开始符号。

素短语的概念：设有文法 G[S]，其句型的素短语是一个短语，它至少包含一个终结符，并除自身外不包含其他素短语，最左边的素短语成为最左素短语。

已知文法 G[T]：T →tlel(F)　F →T+FlT

（1）给出句型((t)+T)的短语、直接短语、句柄、素短语和最左素短语——画出语法分析树

（2）构造该文法的算符优先关系表

（3）判断该文法是否是算符优先文法

（4）给出对输入串(t+e)的分析过程

文法的算符优先关系表

	t	e	(	)	+	#
t				>	>	>
e				>		>
(	<	<	<	=	<	
)				>	>	>
+	<	<	<	>	<	
#	<	<	<			=

符号栈	关系	输入串剩余部分	归约串
#	<	(t+e)#	
#<(	<	t+e)#	
#<(<t	>	+e)#	t
#<(<T	<	+e)#	
#<(<T+	<	e)#	
#<(<T+<e	>	)#	e
#<(<T+T	>	)#	T+T =>T+F
#<(F	=	)#	
#<(F)	>	#	(F)
#T			接受

语法制导翻译：语法分析与语义分析穿插进行，语法分析引导语义分析

设有文法 G[S]:S→(L)la　L→L,SlS 给此文法配上语义规则（即语法制导定义），输出配对括号的个数，如对句子(a,(a,a))，输出是 2

采用自下而上进行归约的方式进行语法分析

步骤

1、拓广文法

S’ →S　S →(L)　S →a　L →L1，S　L →S

2、引进语义变量

为每个文法符号添加一个属性“num”，用于记录归约得到该符号时已经配对的括号数

3、设计语义动作

L →S　　L.num=S.num

L →L1,S　　L.num=L1.num+S.num

S →a　　S.num=0

S →(L)　　S.num=L.num+1

S’ →S　　Print(S.num)

4、形成语法制导定义

产生式	语义动作
S’ →S	{Print(S.num)}
S →(L)	{S.num=L.num+1}
S →a	{S.num=0}
L →L_1,S	{L.num=L_1.num+S.num}
L →S	{L.num=S.num}

符号表的信息将在词法分析、语法分析、语义分析、存储分配等过程中陆续填入。符号表的使用有时会延续到目标的运行阶段。

主要作用：检查语义的正确性　　辅助生成代码

”标识符“与”名字“有何区别？

答：在程序设计语言中，一般以字母开头的字母数字序列（有限个字符）都是标识符。当给予某个标志符确切的含义后，该标识符就叫做一个名字。标识符是一个没有意义的字符序列，而名字却有确切的含义，一个名字代表一个存储单元，该存储单元的内容为该名字的值，同时名字还有属性（即类型和作用域等）。例如 area，作为标识符，它没有任何意义，但作为名字，可以表示变量名、函数名等。

符号表的总体组织：把属性完全相同的符号组织在一起；把各种符号都组织在一张表中；以上两种情况的折中

符号表的数据结构：线性、树、散列表

符号表的组织：名字栏、信息栏

运行阶段的存储组织与分配的方案：静态存储分配、动态存储分配[栈式、堆式(delete、new)]

代码优化的原则：等价原则、有效原则、合算原则

优化涉及范围：局部优化、循环优化、全局优化

基本块：程序中一个顺序执行的语句序列，其中只有一个入口和一个出口，入口就是第一个语句，出口就是最后一个语句。对一个基本块来说，执行时只能从其入口进入，从其出口退出。

划分基本块的算法：

求入口语句：程序的第一个语句；能由条件转移语句或无条件转移语句转移到的语句；紧跟在条件转移语句后的语句

对以上求出的每一入口语句构造其所属的基本块，它是由该入口语句到另一入口语句（不包括该入口语句），或到一转移语句（包括该转移语句），或到一停语句（包括该停语句）之间的语句序列组成的。

凡未被纳入某一基本块的语句，都是无效语句，将其删除。

基本块内可进行的优化：删除公共子表达式、删除无用代码、复写传播、合并已知常量

循环优化：强度削减、代码外提、删除归纳变量

可以从哪些层次上对程序进行优化？

答：为获得更优化的程序，可以从各个环节着手。

（1）在源程序级。选择适当的算法和安排适当的实现语句来提高程序的效率。

（2）在中间代码的设计上。考虑产生更高效的中间代码。

（3）在代码优化级。安排专门的优化阶段，进行各种等价变换。

（4）在目标代码级。考虑如何有效地利用寄存器，如何选择指令。