编译原理

第一章 引论

1.编译程序的相关概念

翻译程序：把一种语言程序（源语言程序）等价地转换成另一种语言程序（目标语言程序）的程序。 编译程序：特指把高级语言程序翻译成低级语言程序。 解释程序：不产生目标程序，边解释边执行。

2.编译过程概述

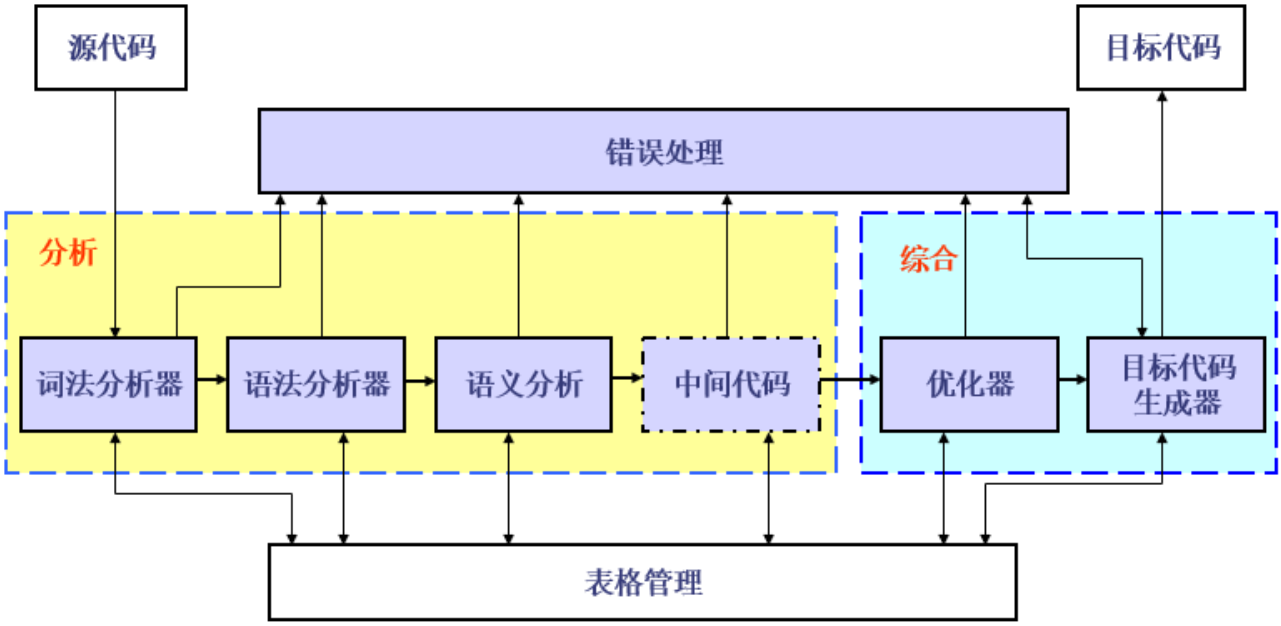
1. 词法分析
2. 语法分析
3. 语义分析与中间代码生成
4. 优化
5. 目标代码生成

3.中间代码的几种表示形式

- **中间代码：**
  - 含义明确、便于处理的记号系统。
  - 独立于具体硬件。
  - 更接近于计算机指令系统或容易变换为机器指令。
  - 三元式，四元式，逆波兰式、树形表示等。
- **例：$id_1+id_2*id_3$**

三元式表示	四元式表示（三地址码）	逆波兰
1. $(*, id_2, id_3)$	1. (*,id2,id3,T1)	$id_1 id_2 id_3 * +$
2. $(+, id_1, (1))$	2. $(+, id_1, T_1, T_2)$	

4.编译的前端与后端（对应图中的分析与综合部分）



第二章 高级语言及其语法描述

1.文法的定义

文法是描述语言语法结构的形式规则（即语法规则）

2.上下型无关文法

上下文无关文法所定义的范畴（语法单位），是完全独立的于这种范畴可能出现的环境的，当遇到一个语法单位可直接对其进行特定的处理而不必考虑它所处的上下文

一个上下文无关文法G包括以下组成，一组**终结符**，一组**非终结符**，一个**开始符号**，一组**产生式**

如果将->替换为::=，那么就是巴科斯范式

3.判断一个文法是否具有二义性的条件

如果某一个文法的句子对应**两棵不同的语法树**，则称这个文法是二义的

4. 0,1,2,3型文法

（1）0型文法：无限制文法，短文文法

特征：产生式左侧至少有一个非终结符

（1）1型文法：也称上下文有关文法

特征：左侧长度小于等于右侧长度，且左侧只有一个非终结符

（2）2型文法：上下文无关文法

- 也称上下文无关文法（CFG：Context-free Grammar)
- 产生式的形式为$P \rightarrow \alpha$, 其中

  $P \in V_N$ ，且$\alpha \in (V_N \cup V_T)^*$

（3）3型文法：也称正规文法

- 也称正规文法
  - 右线性文法（Right-linear Grammar）：

    $A \rightarrow \omega B$ 或 $A \rightarrow \omega$，其中 A、$B \in V_N$, $\omega \in V_T^*$。
  - 左线性文法（Left-linear Grammar）：

    $A \rightarrow B\omega$或 $A \rightarrow \omega$，其中 A、$B \in V_N$, $\omega \in V_T^*$。
  - 对应的语言：正规语言
  - 对应的自动机：有限自动机（Finite Automaton）。
- 例.文法 $S \rightarrow aS$，$S \rightarrow a$

  对应正规式：a^+，或者a^*a

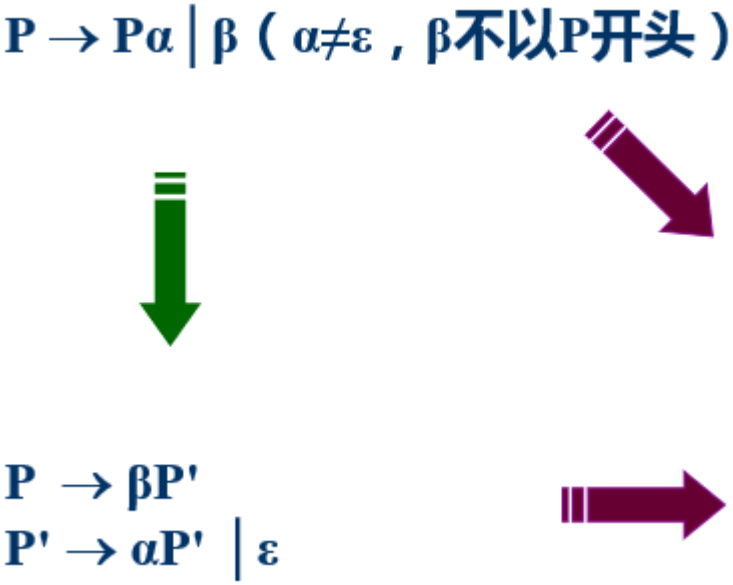
第三章 词法分析

根据文字描述写出正规式以及画DFA，NFA，化简为最简DFA

参照书上P64 9.1

第四章 语法分析—自上而下分析法

1.消除左递归



2.求First,Follow集合

（1）NULLABLE集合

$$X \to \varepsilon$$（直接将X加入$NULLABLE$集合）

$$X \to Y_1 \dots Y_n$$（$Y_1 - Y_n$都属于$NULLABLE$将X加入$NULLABLE$集合）

（2）FIRST集合

$$X \to a$$(直接将a加入$First$集合)

$$X \to Y_1 \dots Y_n (FIRST（X）\cup FIRST（Y_1）;$$
当Y_1属于$NULLABLE$时，选择下一个，直到选到不属于$NULLABLE$的为止，
将此集合的$FIRST$集合并入$FIRST（X）$

（3）FOLLOW集合

如S->(L)｜aL｜LC
找Follow的三种情况：先在候选式（右边）中找到该非终结符，如L（注意例中只有一个定义，但找Follow要看到所有右边出现该非终结符的）　　如果L的右边是终结符，　那么这个终结符加入L的Follow　　如果L的右边是非终结符，那么把这个非终结符的First除去空加到L的Follow中　　如果L处在末尾，　　那么，'->'左边符号的Follow成为L的Follow 另外要注意的是：　　开始符号的Follow中要加上'#'
技巧：Follow一般从上往下找。　　如果要找L的Follow，要从式子的右边找到L，然后来找L的Follow，这与First是不同的。

4. LL（1）文法的判断

- 如果文法G满足以下条件：

（1）消除了左递归；

（2）各个产生式的候选首符集两两不相交，即

若 $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ ，

则 $FIRST(\alpha_i) \cap FIRST(\alpha_j) = \Phi$ ，（$i \neq j$）；

（3）对文法中的每个非终结符A属于NULLABLE，则

$FIRST(A) \cap FOLLOW(A) = \Phi$ ，

称该文法G为LL(1)文法。

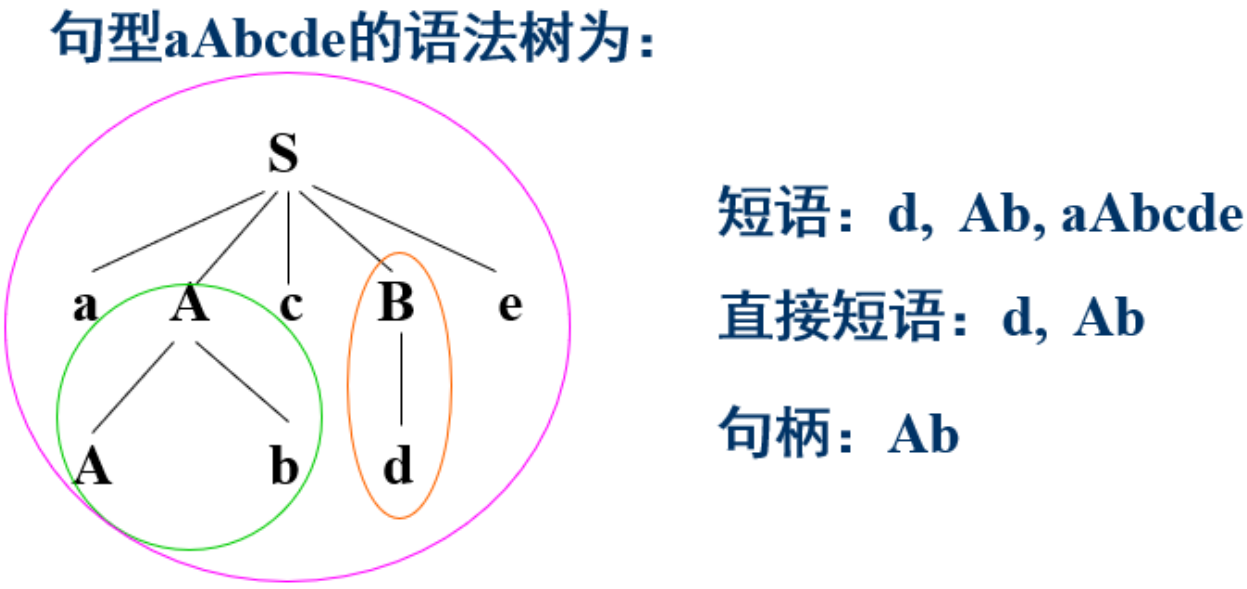
第五章—自下而上分析法

1.自下而上分析法的概念

所谓自下而上分析法就是从输入串开始，逐步进行规约，直到规约到文法的开始符号为止

2.直接短语与句柄

语法树中的叶子节点称为直接短语，最左侧的直接短语称为句柄，如下图所示：



3.LR（0）项目

- 文法G的产生式加上一个位于产生式体中某处的圆点(·)，称为G的一个LR(0)项目(项，item)。
- 例：A → BC
  - A → · BC
  - A → B · C
  - A → B C·
- 产生式A→ε只对应一个项目A→·
- 可以用一对整数表示，第一个是产生式的编号，第二个是点的位置。项指明了语法分析过程中的给定点上，我们已经看到了产生式的哪一个部分。

LR(0)文法的判断

- 假若一个文法G的拓广文法G'的活前缀识别自动机中的每个状态（项目集）不存在下述情况：
  - 既含移进项目又含归约项目；
  - 含有多个归约项目
- 则称G是一个LR(0)文法。

SLR(1)文法的判断依据

对产生冲突的项目集进行判断，若FOLLOW(A)∩FOLLOW(B)结果为空集，则说明此冲突可以解决，此文法为SLR（1）文法

第六章—属性文法与语法制导翻译（只考察几个概念）

综合属性：此节点的属性只由其子节点决定

继承属性：节点的属性由父节点或兄弟节点决定

S-属性文法：只含有综合属性

L-属性文法：对于每个产生式A->X1X2X3...Xn，每条语义规则计算的属性是A的综合属性；或者是Xj 的继承属性，1 ≦j ≦ n, 但它仅依赖：

1. 该产生式中Xj左边符号X1, X2, ..., Xj-1的属性
2. A的继承属性