编译原理

by xd_zhu

笔记

零、考试

- 考试时间

> 7.9 9:00-11:00

- 答疑时间

> 7.8 全天 C109

- 题型

> ① 填空题（20）
> ② 简答题（20）
> ③ 计算题（60）

- 重点

> ① 编译器的功能、编译的阶段、编译的遍
> ② 词法分析的输出、给定正规式 → 描述的语言、有限的规则 → 无限的符号串
> ③ 描述的语言或集合 → 正规式、正规式 → NFA → DFA → 最小化DFA
> ④ CFG和CFL概念、给定产生式 → 描述的语言、判断二义性
> ⑤ 预测分析法（推导、匹配、回溯、接受、报错）、LR分析法（移进、归约、接受、报错）
> ⑥ 消除左递归、构造预测分析表、判断LL(1)文法、构造SLR(1)分析表，判断冲突
> ⑦ 继承属性、综合属性、语义动作（语法分析时匹配或规约的时候）、参数传递
> ⑧ 给定语句 → 三地址码、产生式+语义规则 → 注释语法树

一、绪论

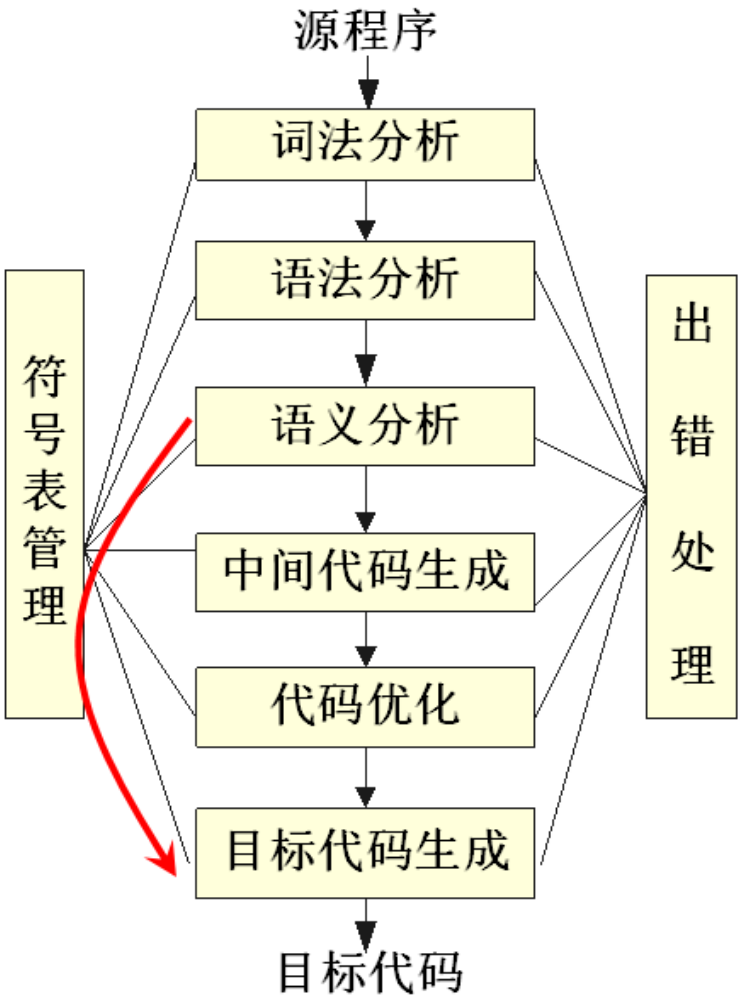
- 编译器

> 把一种语言程序（源程序）转换成另一种语言程序（目标程序）

- 编译的遍

> 对源程序或源程序的中间结果从头到尾扫描一次，并做有关的加工处理，生成新的中间结果或目标程序

- 编译器工作原理



二、词法分析

- 词法分析器

> ① 滤掉源程序中的无用成分，如注释、空格、回车等
> ② 处理与平台有关的输入，如文件结束符的不同表示等
> ③ 根据模式识别记号，并交给语法分析器
> ④ 调用符号表管理器或出错处理器，进行相关处理
>
> 输出：记号序列
> 工作方式：单独扫描，作为语法分析器的子程序，并行工作，使用队列

- 有限自动机（FA）

> $$M=(S,\sum,move,s_0,F)$$
>
> ① 非确定型有限自动机（NFA）：不确定性，大量回溯
> ② 确定型有限自动机（DFA）：没有ε状态转移，每个s_i和a最多一个下一状态

- 正规式构造NFA（Thomspson算法 或 经验）

> 先分解、再合并，每一步最多引入两个新状态，P26

- NFA构造DFA

> ① $\varepsilon\_CLOSURE(I)$：从状态I出发，不经任何字符达到的状态全体
> ② $smove(S,a)$：从状态集S出发，标记为a的下一状态全体
> ③ NFA构造DFA：P30

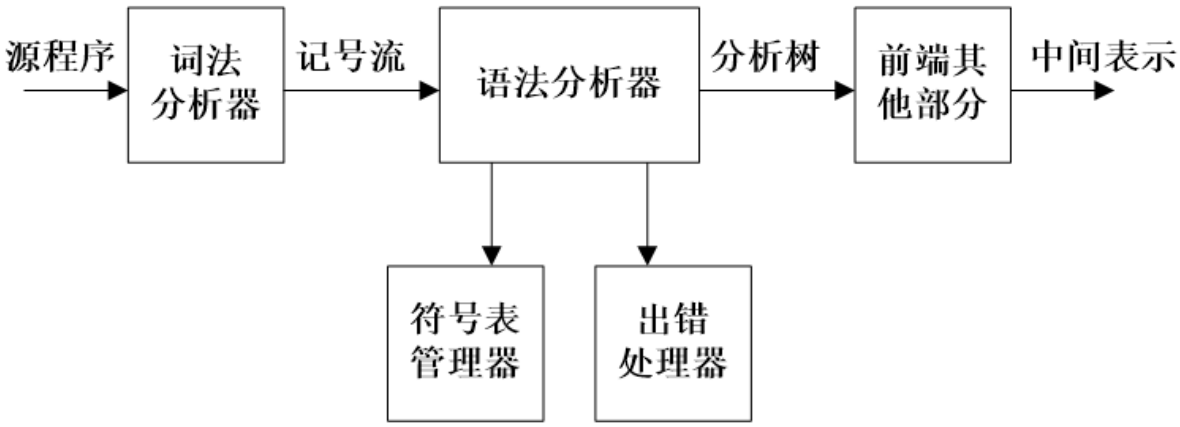
- 最小化DFA

> 在状态集S上找到一个基于不可区分的等价关系R，S/R即是最小DFA的状态集，P32

三、语法分析

- 语法分析器

> 作用：构造分析树、检查语法错误



- 上下文无关文法（CFG）

> 上下文无关语言（CFL）

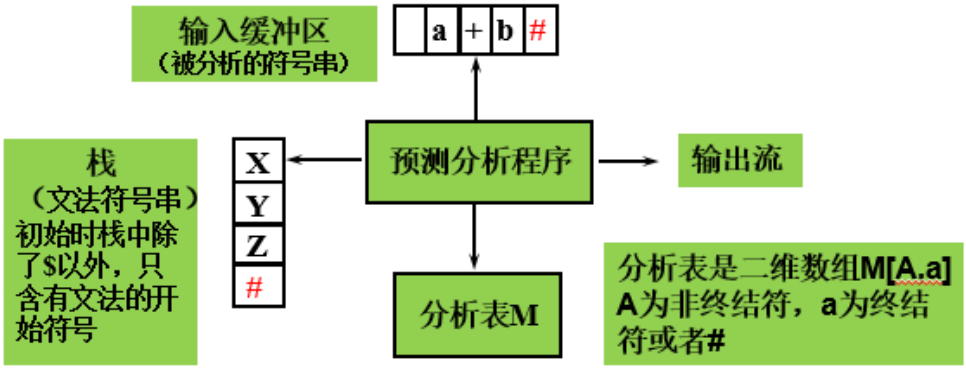
- 二义性

> 定义：一个句子可能对应多余一颗分析树
> 原因：文法符号缺乏优先级和结合性的规定
> 消除方法：①改写为非二义文法，②规定符号的优先级和结合性，③修改语言的语法
>
> 改写方法：
> ① 引入新的非终结符，增加一个子结构并提高一级优先级
> ② 递归非终结符在终结符左边，运算具有左结合性，否则具有右结合性

- 自上而下语法分析

> 策略：最左推导，从左到右扫描，自上而下建分析树，试探
> 方法：递归下降分析法、预测分析法
> 问题：①公共左因子 $\rightarrow$ 大量回溯，效率低，②左递归 $\rightarrow$ 死循环
>
> 先消除左递归，再提取左因子
> ① 消除直接左递归 $A \rightarrow A\alpha$
> $A \rightarrow A\alpha_1|A\alpha_2|\ldots|Aa_m|\beta_1|\beta_2|\ldots|\beta_n$（ α_i非空，β_i不以A开始）
> 替换为：
> $A \rightarrow \beta_1 A'|\beta_2 A'|\ldots|\beta_n A'$
> $A' \rightarrow \alpha_1 A'|\alpha_2 A'|\ldots|\alpha_n A'|\varepsilon$
> ② 消除间接左递归 $A \overset{+}{\rightarrow} A\alpha$
> 核心思想：将不是直接左递归的符号右部展开到其他产生式，再消除直接左递归
> ③ 提取左因子
> $A \rightarrow \alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n|\gamma$
> 替换为：
> $A \rightarrow \alpha A'|\gamma$
> $A' \rightarrow \beta_1|\beta_2|\ldots|\beta_n$

- 预测分析法



> 组成：预测分析表、表驱动程序、分析栈
> 动作：推导、匹配、回溯、接受、报错
>
> $FIRST(\alpha)=\{a \mid \alpha \overset{*}{\Rightarrow} a\ldots, a\in T\}$，若$\alpha \overset{*}{\Rightarrow} \varepsilon$，则$\varepsilon \in FIRST(\alpha)$
> 含义：从α开始可以推导出的所有开头终结符
> ① 若$X\in T$，则$FIRST(X)=\{X\}$
> ② 若$X\in N, X\to a\alpha$，则$a\in FIRST(x)$，若$X\to\varepsilon$，则$\varepsilon\in FIRST(X)$
> ③ 若$X\in N, X\to Y_1Y_2, \varepsilon\in FIRST(Y_1)$，则$FIRST(Y_2)\subseteq FIRST(X)$
>
> $FOLLOW(A)=\{a \mid S\overset{*}{\Rightarrow}\ldots Aa\ldots, a\in T\}$
> 含义：从开始符号可以推导出的所有紧跟A之后的终结符
> ① $\#\in FOLLOW(S)$，若A是某句型的最右符号，则$\#\in FOLLOW(A)$
> ② 若$A\to\alpha B\beta$，则$FIRST(\beta)-\{\varepsilon\}\subseteq FOLLOW(B)$
> ③ 若$A\to\alpha B$ 或 $A\to\alpha B\beta, \varepsilon\in FIRST(\beta)$，则
> $FOLLOW(A)\subseteq FOLLOW(B)$
>
> 构造预测分析表
> ① 对每个产生式$A\to\alpha$，其$FIRST(A)$的每个终结符a，把$A\to\alpha$ 加入到$M[A,a]$中
> ② 对每个产生式$A\to\alpha$，若$\varepsilon\in FIRST(\alpha)$，则对$FOLLOW(A)$的每个终结符$b$，把$A\to\varepsilon$加入到$M[A,b]$中

- LL(1)文法

> 判断方法1：预测分析表中不含多重定义的条目
> 判断方法2：任何两个产生式$A\to\alpha|\beta$满足
> ① $FIRST(\alpha)\cap FIRST(\beta)=\phi$
> ② 若$\beta\overset{*}{\Rightarrow}\varepsilon$，则$FIRST(\alpha)\cap FOLLOW(A)=\phi$

- 自下而上语法分析

> 策略：从左到右扫描ω，反复用产生式的左部替代右部，剪句柄
> 方法：算符优先分析、移进-规约分析法
> 问题：①如何确定句柄，②如何选择正确的产生式

- SLR(1)

> 组成：移进-归约分析表、驱动器、符号状态栈
> 动作：移进、归约、接受、报错
>
> $CLOSURE(I)$：项目集I的闭包
> 含义：从项目集I不经任何文法符号到达的项目全体
> ① $I \subseteq CLOSURE(I)$
> ② 若$A \to \alpha \cdot B\beta \in CLOSURE(I)$，则$B \to \cdot\gamma \in CLOSURE(I)$
>
> $GO(I,X)$：状态转换函数
> 含义：所有从I经文法符号X能到达的项目全体
> ① $J = \{A \to \alpha X \cdot \beta \mid A \to \alpha \cdot X\beta \in I\}$
> ② $GO(X,I) = CLOSURE(J)$
>
> ① 移进-归约冲突：$X \to \alpha \cdot b\beta, A \to \alpha \cdot$
> ② 规约-归约冲突：$A \to \alpha\cdot, B \to \alpha \cdot$
> 解决:
> 一个项目集含 $\begin{cases} A_1 \to \alpha \cdot a_1\beta_1, A_2 \to \alpha \cdot a_2\beta_2, \ldots, A_m \to \alpha \cdot a_m\beta_m \\ B_1 \to \alpha\cdot, B_2 \to \alpha\cdot, \ldots, B_n \to \alpha \cdot \end{cases}$
> 且
> $\{a_1, a_2, \ldots, a_m\} \cap FOLLOW(B_1) \cap FOLLOW(B_2) \cap \ldots \cap FOLLOW(B_n) = \phi$

SLR(1)分析表的构造　　(Follow集合可从DFA中看出)

```
for 每个状态转移Dtran[i,x] == j:
    if x是终结符:
            action[i,x] = Sj;
    else x不是终结符:
        goto[i,x] = j;
for 每个状态i的每个可归约项A→α.:
    if  S'→ S.:
        action[i, #] = acc;
    else:
        for 每个a∈FOLLOW(A):
            action[i,a] = Rk;    (k为A→α在原始文法的第几条)
```

四、语法制导翻译与中间代码生成

- 语义分析

> 作用：检查语义合法性、规定语义动作
> 方法：语法制导翻译

- 属性文法

> $$b := f(c_1, c_2, \ldots, c_k)$$
>
> ① 综合属性：自下而上
> ② 继承属性：自上而下

- 语义规则

> ① 语法制导定义（设计）
> ② 翻译方案（实现）

- LR分析翻译

> ① 扩充LR分析器：增加产生式对应的语义动作
> ② 扩充分析栈：增加并列于分析栈的语义栈

- 中间代码

> 特性：既与机器指令的结构相近，又与具体机器无关
> 作用：便于编译器的开发移植和代码优化
> 形式：
> ① 后缀式
> ② 三地址码：三元式、四元式
> ③ 图形表示：抽象语法树（后序遍历）、DAG