编译原理复习提纲（上）

文法、词法分析（NFA/DFA）、语法分析（LL(1)、OG、LR(0)、SLR(1)、LR(1)、LALR(1)）

# 编译程序模型

输入：源程序

分析阶段：词法分析、语法分析、语义分析

综合阶段：中间代码优化、优化、目标代码生成器

两个阶段均有：错误处理、符号表管理

输出：目标代码

## 词法分析

内码：二元式

## 语法分析

语法规则用BNF表示

## 语义分析和中间代码生成

中间代码常用：逆波兰式、三元式、四元式、抽象机代码表示

## 代码优化

等价变换原则

## 目标代码生成

绝对指令代码→可重定位指令代码或汇编指令代码

## 表格管理

常见符号表：名字特性表、常数表、标号表、分程序入口表、中间代码表

## 前端和后端

前端：编译逻辑结构中的分析部分

后端：与目标机器有关的部分

# 文法和语言

## 产生式

## 文法

## 句型

## 规范推导

最右推导为规范推导，规范推导产生的句型为规范句型。

## 递归规则

### 直接左递归的产生式

### 间接左递归的产生式

### 递归文法

至少含一个递归的非终结符号。

## 短语

有句型：

### 简单（直接）短语

### 句柄

句型的**最左**简单短语

## 上下文无关文法（2型、CFG）

左边只有一个非终结符。

## 正则文法（3型）

## CFG的化简

### 消除无用符号

#### 可达

#### 产生

#### 有用

若X同时是产生的和可达的，则称X是有用的，否则为无用符号。

#### 消除算法

##### 计算“产生的”符号集算法

1. 每个T中的符号都是产生的
2. 若有产生式且中符号都是产生的，则A是产生的

##### 计算“可达的”符号集算法

1. 符号S是可达的
2. 若有产生式且是可达的，则中的符号都是可达的

##### 示例

消除非产生的

消除非可达的

必须先消除非产生的，再消除非可达的。

### 消除

形如的产生式为空产生式或。

#### 算法

先确定全部可空的变元：

1. 若，则A可空
2. 若且中每个符号都是可空的，则B可空

再替换带可空符号的产生式，若是产生式，那么用所有的替代，其中：

1. 若不是可空的，则
2. 若是可空的，则或
3. 但不能全部为

#### 示例

消除后

### 消除单元产生式

形如。

代入消除即可。

### 简化的可靠顺序

1. 消除
2. 消除单元产生式
3. 消除非产生的无用符号
4. 消除非可达的无用符号

# 词法分析

## 有穷自动机

### 确定的有穷自动机（DFA）

### 非确定的有穷自动机（NFA）

### NFA→DFA

#### move

#### 算法

计算起始状态的，记为，依次计算。

### DFA最小化

标记状态对。（即终点和非终点不能为一类）

若被标记了，则标记，重复该步骤，直到无可标记。（即目的地不在一类的点对不在一类）

#### 定理

对于有同一接受集的FA，与之等价且具有最小状态数的DFA在同构意义下是唯一的。

## 正则式（正规式）

与FA等价。

# 自顶向下语法分析

## 下推自动机

## LL(1)文法

### 构造

#### FIRST集（开头集，可能的开头的终结符集合）

若，则

#### FOLLOW集（跟随集，后面可能跟随的终结符集合）

若，则

#### SELECT集（产生式预测集）

### 判别

相同左部的产生式的集的交集都为空，则是LL(1)文法。

### 非LL(1)转换为LL(1)

* 提取左公共因子
* 消除左递归
  + 直接左递归
  + 间接左递归

### 预测分析表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

#### 分析过程

|  |  |  |  |
| --- | --- | --- | --- |
| 步骤 | 分析栈 | 剩余输入串 | 所用产生式或匹配 |
| *1* |  |  |  |
| *2* |  |  |  |
| *3* |  |  |  |
| *4* |  |  |  |
| *5* |  |  |  |
| *…* | *…* | *…* | *…* |

# 自底向上优先分析

## 算符优先分析法

### 优先关系

### 构造优先关系矩阵

#### 集

#### 集

#### 算法

对每个产生式

1. 若和都是终结符：
2. 若是终结符是非终结符：

规定#优先级比相邻任何运算符都低。

不可以同时出现任意两种。

### 素短语

* 至少包含一个终结符
* 除他自身，不再包含其他素短语

#### 最左素短语

满足

### 规约过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 栈 | 优先关系 | 当前符号 | 剩余输入串 | 移进或规约 |
|  |  |  |  | 移进 |
|  |  |  |  | 规约 |
|  |  |  |  | 移进 |
|  |  |  |  | 移进 |
|  |  |  |  | 规约 |
|  |  |  |  | 规约 |
|  |  |  |  | 接受 |

## LR(0) 分析

### 拓广文法

增加产生式

### 活前缀

规范推导（最右推导）：

其逆过程为最左规约（规范规约）。

每次规约前句型的前部，称为可归前缀。

把形成可归前缀之前，包括可归前缀的所有前缀，称为活前缀。

#### 计算不包含句柄的活前缀

即，若有产生式

则

#### 计算包含句柄的活前缀

#### 例

求不包含句柄在内的活前缀方程组：

所以包含句柄的活前缀为：

### 项目集规范族

#### LR(0) 项目

在产生式的右部每个空隙加一个圆点。

#### 构造NFA/DFA

#### LR(0) 分析表构造

*若*

*若*

*若*

最终表形如：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状态 |  | | | | |  | | |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

### LR(0) 分析器工作

如上例中的分析表，分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 状态栈 | 符号栈 | 输入串 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

遇到规约后，状态栈弹出个状态，即第个产生式右部符号数。

然后以此时栈顶状态为当前状态，当前输入为规约后的非终结符，找到对应的。

## SLR(1) 分析

### 基本思路

利用非终结符号的FOLLOW集，判断“规约”还是“移进”。

### 解决冲突

若LR(0)的规范族含有如下项目集

存在移进-移进冲突和移进-规约冲突。

若

则当状态面临输入符号时，

## LR(1) 分析

### LR(1) 项目集族的构造

起始项目为

### LR(1) 分析表构造

*项目*

其余不变

## LALR(1) 分析

### 基本思路

合并同心集而不产生冲突。超前搜索符为之前的合集。

## 优先关系解决二义性文法

例如状态：

由于，所以遇移进，而符合左结合，所以遇规约。

又例如状态：

由于，所以无论如何都规约。