第四章 语法分析 →∪Σε

4.3 设计文法

4.3.1 词法分析和语法分析

任何正则表达式都可以使用文法来描述。为什么使用正则表达式来定义一个语言的词法？

1) 将一个语言的语法结构分为词法和非词法两部分，可以很方便地将编译器前端模块化，将前端分解为两个大小适中的组件；

2) 一个语言的词法规则通常很简单，我们不需要使用像文法这样的功能强大的表示方法来描述这些规则；

3) 和文法相比，正则表达式通常提供了更加简洁且易于理解的表示词法单元的方法；

4) 根据正则表达式自动构造得到的词法分析器的效率要高于根据任意文法自动构造得到的分析器。

4.3.2 消除二义性

一个文法，如果存在某个句子不止一棵分析树，或者说这个句子存在不止一种最左（最右）推导，那么称这个文法是二义的。

a = b = c

消除以下文法的二义性

E → E = E | **id**

letter -- > **id**

expr -- > letter = expr | letter

例(1) ： E → E + E | E – E | 0 | 1 | 2 | … | 9

term -- > digit

expr --> expr + term | expr – term | term

term指一个基本操作数

求 9 – 5 + 2 的分析树

可以得到两种分析树: (9-5)+2和9-(5+2).

该文法没有考虑运算符的结合性， 考虑到+和-都是左结合, 即 9 – 5 + 2, 先计算左边-, 再计算右边+,

9 + 5 + 2

+ - \* /

基本操作数: () digit

\* / 左结合

+ - 左结合

*expr* -- > expr + term | expr – term | term

*term* -- > term \* factor | term / factor | factor

*factor* -- > (*expr*) | digit 基本操作数

9 + 5 \* 2

左结合：先计算左边，使用左递归文法

重新设计文法:

expr → expr + term | expr – term | term

term → 0 | 1 | … | 9

如 9 – 5 + 2

expr ==> expr + term == > expr – term + term == >\* 9 – 5 + 2

如果是右结合呢??

如 a = b = c; 其中=号是右结合运算符. 使用右递归文法

expr → letter = expr | letter

letter → a | b | … | z

例(2) :

9 + 5 \* 2

优先级: \* / > + -

并且 \* / + - 都是左结合

假设expr用于表示所有算术表达式

先考虑表达式的基本单元, 可以是一个数字，也可以是一个()表达式

factor → **digit** | ( expr )

先考虑优先级高的 \* /

term → term \* factor

term → term / factor

term → factor

再考虑优先级低的 + / :

expr → expr + term

expr → expr – term

expr → ter

*expr* → *expr* + *term*

| *expr* -*term*

| *term*

*term* → *term \* factor*

| *term / factor*

| *factor*

*factor* → **id**

| (*expr*)

练习(1) ：S →S and S | S or S | not S | p | q | (S)

假设 优先级从低到高排列：or，and，not

并且or, and都是左结合的, not是右结合

消除该文法的二义性.

(1) 设计基本操作数的产生式 A -- > p | q | ( D )

(2) 设计运算符not的产生式 B -- > not B | A   
(3) 设计运算符and的产生式 C -- > C and B | B

(4) 设计运算符or的产生式 D --> D or C | C

例(3) 悬空else文法

*stmt* → **if** *expr* **then** *stmt*

| **if** *expr* **then** *stmt* **else** *stmt*

*|* **other**

写法 **if** *expr* **then if** *expr* **then** *stmt* **else** *stmt* 两个不同推导.

同时带if/else的条件语句 matched; 只带if, 不带else的条件语句 unmatched

*stmt* → *matched\_stmt*

| *unmatched\_stmt*

*matched\_stmt* → **if** *expr* **then** *matched\_stmt* **else** *matched\_stmt*

*|* **other**

*unmatched\_stmt* → **if** *expr* **then** *stmt*

| **if** *expr* **then** *matched\_stmt* **else** *unmatched\_stmt*

4.3.3 消除左递归

如果一个文法是左递归的，如果它有非终结符A，对某个串α，  
存在推导A => + Aα.

由形式为*A→Aα*的产生式引起的左递归称为**直接左递归 (立即左递归)**

A → Aα | β

A描述的是: 以β开始，后面接0个多个α

A → β A'

A' → α A' | e

更一般情况:

A → Aα1 | Aα2 | … | Aαm | β1 | β2 | … | βn

A → β1A' | β2 A' | … | βn A'

A' → α1A' | α2 A' | … | αm A' | e

非直接左递归

S → Aa | b

A → Sd | ε

A → Aad | bd | ε

A → bdA' | A'

A' → ad A' | ε

其中非终结符S是左递归的，S => Aa => Sda，但它不是直接左递归的。间接左递归

如何消除非直接左递归?

4.3.4 提取左公因子

*stmt* → **if** *expr* **then** *stmt* **else** *stmt*

| **if** *expr* **then** *stmt*

左公因子 **if** *expr* **then** *stmt*

左公因子会引起的问题: 预测分析或自顶向下分析时，当下一个待分析词法单元为 **if**时，不清楚应该选择哪个产生式.

A → αβ1 | αβ2

消除左公因子α

A → αA'

A' → β1 | β2

一般情况:

A → αβ1 | αβ2 | … | αβn | r

A → αA' | r

A' → β1 | β2 | … | βn