

编译原理实验二报告



2016-11-16

南京大学软件学院

张云飞 141250197

## 实验题目

语法分析程序的代码实现

## 实验内容

根据设计的上下文无关文法，提取公共左因子和消除左递归后，利用程序生成LL(1)文法的预测分析表，然后对输入的token序列进行语法检测。

## 实验步骤

1. 设计上下文无关文法

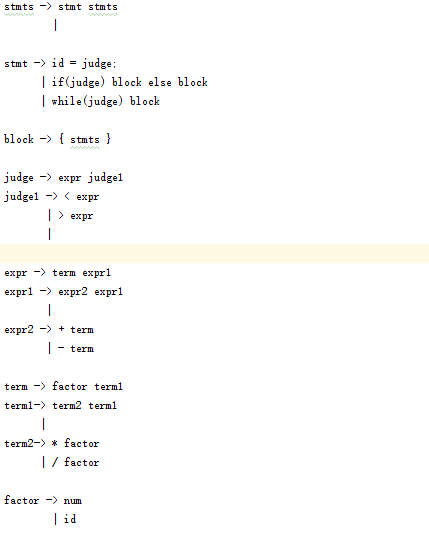
文法支持以下表达式：

a)普通的加减和赋值表达式

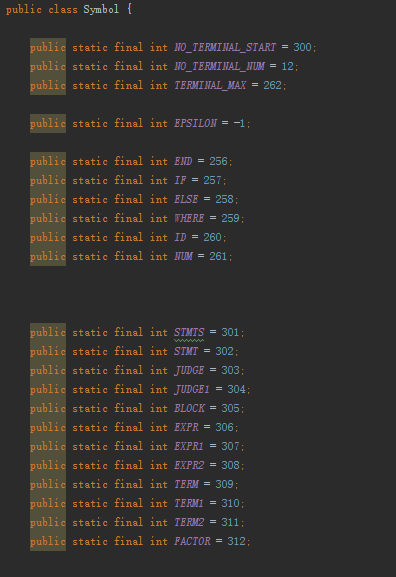
b)if else 语句

c)While 语句

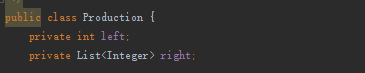
对文法做提取公共左因子和消除左递归后的结果如下：



1. 根据文法自动生成LL(1)的预测分析表
2. 首先将文法中的终结符和非终结符映射为一个对应的数值。在Symbol类中实现符号与数值的映射。

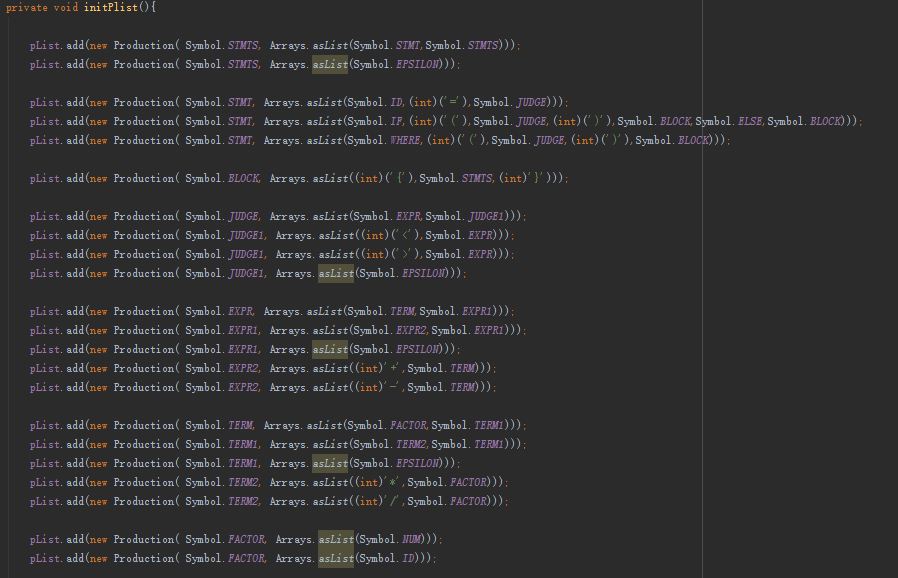


1. 将文法表达式转为数值表示形式写入程序中



定义Production类，左侧为非终结符，右侧为非终结符推导的右部。

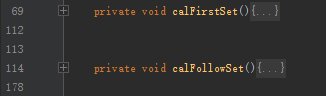
在ProductonTable类中把之前设计好的文法表达式写入。



1. 计算first和follow集合



在ProductionTable中为每个非终结符建立first和follow的映射集合。First和follow集合的计算方式按照课件中的流程通过有限次的循环最终生成。



1. 生成预测分析表



在PPTBuilder中建立二维数组保存预测分析表。根据之前计算得到的first,follow集合和文法推到表达式，最终计算生成预测分析表。

1. 根据生成的预测分析表对输入进行语法推导分析



在类ProductionHandler中，利用在parserStack上的一系列操作，完成对语法的推导分析。

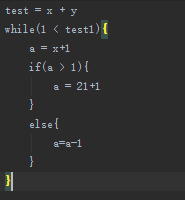
## 补充说明

1. token序列的生成

利用实验一中的程序，在source.txt中输入语句，实验一中的代码会将输入语句转化为一个token序列，提供给语法分析程序使用。

1. 输入和输出

输入放在source.txt中，当前的输入为：



可以按照当前格式输入运算表达式，if,else和while语句，目前的文法中定义每个if后面必须有一个else，如果没有会提示错误。

输出在控制台下，首先会输出定义的正则表达式，然后输出解析后的token序列。最后输出语法的推导解析过程。当前输入下，推导解析过程如下：

自顶向下语法分析过程：

stmts->stmt stmts

stmt->id = judge

judge->expr judge1

expr->term expr1

term->factor term1

factor->id

term1->epsilon

expr1->expr2 expr1

expr2->+ term

term->factor term1

factor->id

term1->epsilon

expr1->epsilon

judge1->epsilon

stmts->stmt stmts

stmt->while ( judge ) block

judge->expr judge1

expr->term expr1

term->factor term1

factor->num

term1->epsilon

expr1->epsilon

judge1->< expr

expr->term expr1

term->factor term1

factor->id

term1->epsilon

expr1->epsilon

block->{ stmts }

stmts->stmt stmts

stmt->id = judge

judge->expr judge1

expr->term expr1

term->factor term1

factor->id

term1->epsilon

expr1->expr2 expr1

expr2->+ term

term->factor term1

factor->num

term1->epsilon

expr1->epsilon

judge1->epsilon

stmts->stmt stmts

stmt->if ( judge ) block else block

judge->expr judge1

expr->term expr1

term->factor term1

factor->id

term1->epsilon

expr1->epsilon

judge1->> expr

expr->term expr1

term->factor term1

factor->num

term1->epsilon

expr1->epsilon

block->{ stmts }

stmts->stmt stmts

stmt->id = judge

judge->expr judge1

expr->term expr1

term->factor term1

factor->num

term1->epsilon

expr1->expr2 expr1

expr2->+ term

term->factor term1

factor->num

term1->epsilon

expr1->epsilon

judge1->epsilon

stmts->epsilon

block->{ stmts }

stmts->stmt stmts

stmt->id = judge

judge->expr judge1

expr->term expr1

term->factor term1

factor->id

term1->epsilon

expr1->expr2 expr1

expr2->- term

term->factor term1

factor->num

term1->epsilon

expr1->epsilon

judge1->epsilon

stmts->epsilon

stmts->epsilon

stmts->epsilon

语法分析完成