Assignment2

ID	2019K8009929039
name	周鹏宇

练习2.1.1：考虑下面的上下文无关文法：

S→ +SS | *SS | a

1) 试说明如何使用文法生成串*a+aa

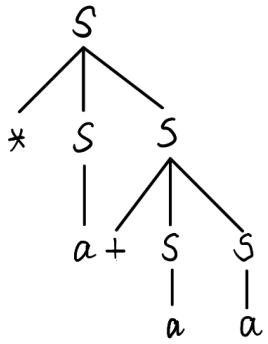
2) 试为这个串构造一颗语法分析树

3) 该文法生成的语言是什么？

4) 该文法具有二义性吗？为什么？

1). $S \rightarrow *SS \rightarrow *aS \rightarrow *a+SS \rightarrow *a+aS \rightarrow *a+aa$

2).



3). 该文法生成的语言为包含+和*运算以及操作数a的前缀表达式

- 在排除只有操作数的式子外，对于任意上述语言的串，都可以由$+st$或$*st$的形式表示，其中子串s，t都可以递归的由上式表示，最终必然能递归到$+a^*_1a^*_2$或$*a^*_1a^*_2$的形式，而对于上述形式的串，可以由S派生，则再逐层退出递归即可派生出目标串，故其可以生成任何前缀表达式

- 由前缀表达式的定义，显然此文法所推出的任意字符串都是包含+和*运算以及操作数a前缀表达式

4). 不具有。不妨假设经过至少n次推导，所产生的字符串s会产生二义性；则显然s不为a，则s必具有$+s_1s_2$或$*s_1s_2$的形式，不妨以$+s_1s_2$为例，若要得到此形式，第一步必然为$S \to +SS$，则推导出s_1, s_2的次数必然小于n次，产生矛盾，故不具有二义性。

练习2.1.2：考虑文法

num → 101 | 1111 | num 0 | num num

1) 证明：用该文法生成的所有二进制串的值都能被5整除（提示：对语法分析树的结点数目，即推导步数，使用数学归纳法）

2) 上面的文法是否能够生成所有能被5整除的二进制串？

1). 不妨假设对于任意推导次数n，此命题都成立，则有：

- 对于$n = 1$，有且仅有101和1111（即5和15）两种情况，自然可以被5整除
- 假设对于$n \leq k$的情况，此命题都成立，则对于$n = k + 1$的情况，有
  - 若第$k+1$次推导为$num \to num_1 0$的情况，则对于num_1，经过k步推导可以得到串s，由假设可知，串s可以被5整除，则$s0 =$（十进制下）$2 * s$必然也可以被5整除
  - 若第$k+1$次推导为$num \to num_1 num_2$的情况，则对于num_1，经过k步推导可以得到串s_1，由假设可知，串s_1可以被5整除；对于num_2，经过k步推导可以得到串s_2，由假设可知，串s_2可以被5整除；则此时$s_1s_2 =$ （十进制下）$2^{digit(s_2)} * s_1 + s_2$可以被5整除

  即对于$n = k + 1$的情况也成立

综上，此命题对于任意的自然数次推导成立

2). 对于0而言，0可以被5整除但无法由此文法生成。

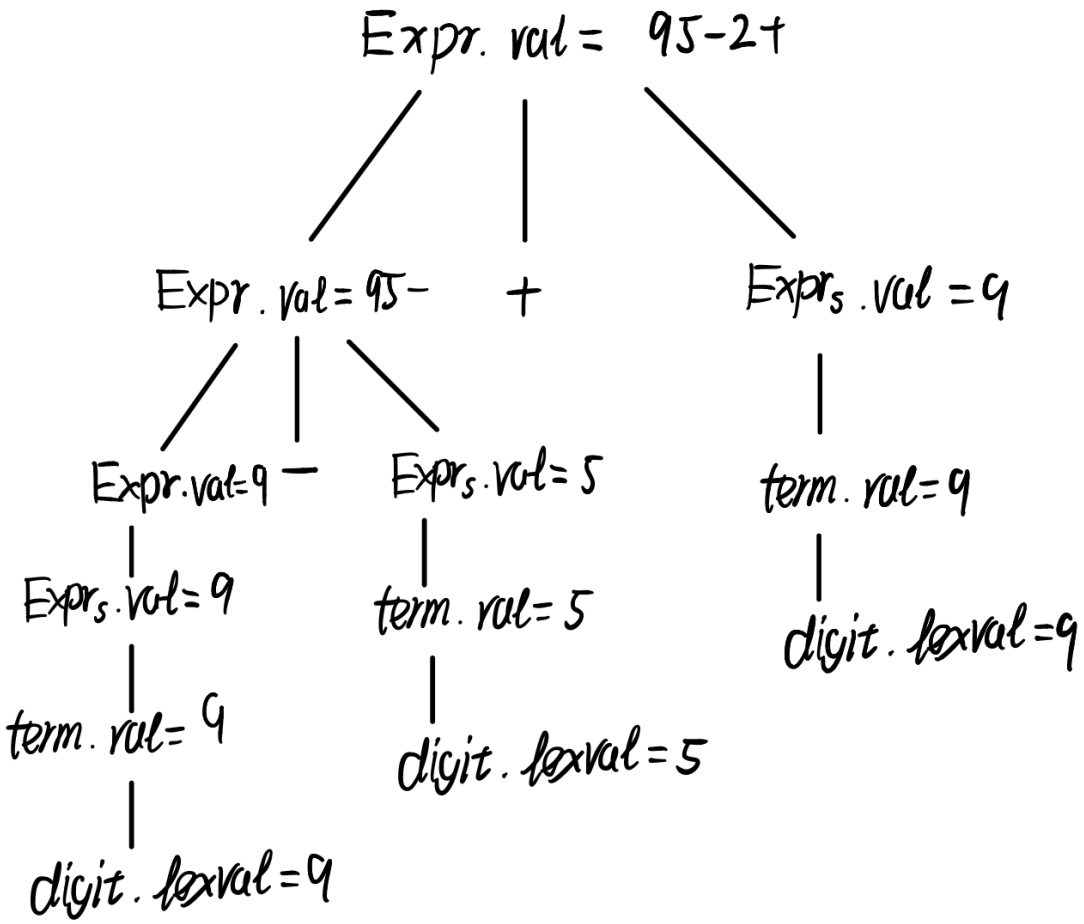
练习2.1.3：构建一个语法制导翻译方案，该方案把算术表达式从中缀表示方式翻译为运算符在运算分量之后的后缀表示方式。例如，xy−是表达式x−y的后缀表示。给出输入9−5+2和9−5∗2的注释分析树。

若考虑乘除优先级，则应有：

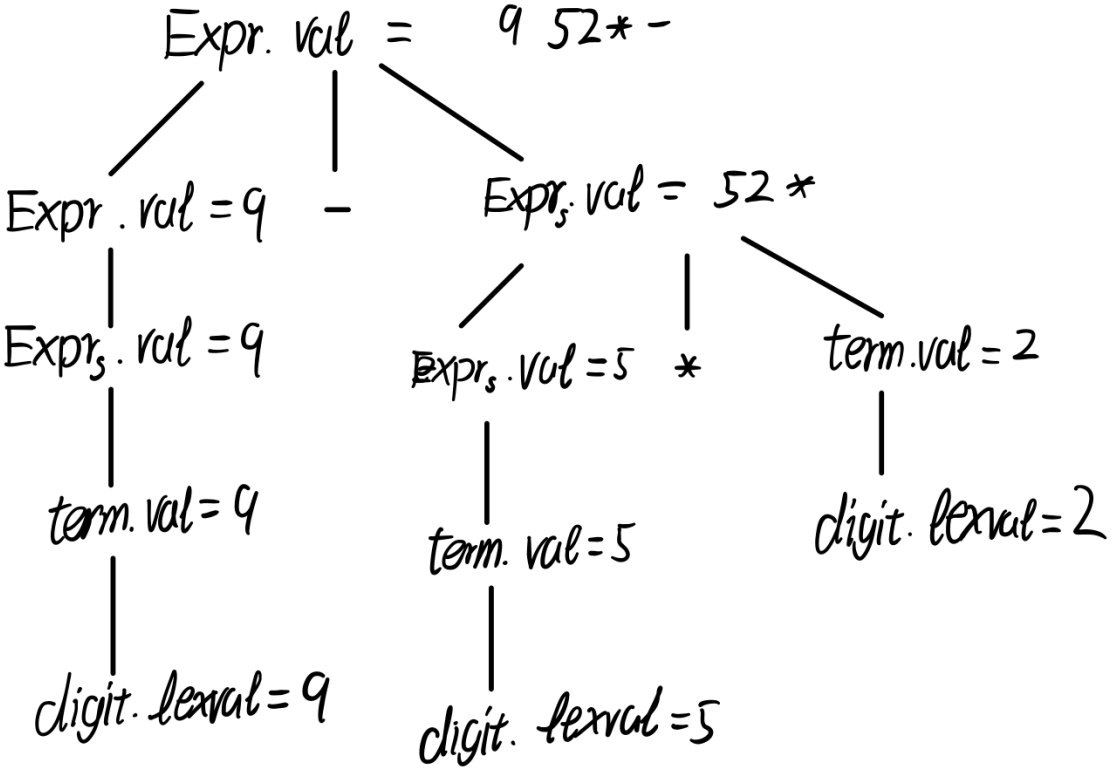
$$
\begin{gathered}
expr \rightarrow expr_1 + expr_s\{print('+')\} \\
expr \rightarrow expr_1 - expr_s\{print('-')\} \\
expr \rightarrow expr_s \\
expr_s \rightarrow expr_{s_1} * term\{print('*')\} \\
expr_s \rightarrow expr_{s_1} / term\{print('/')\} \\
expr_s \rightarrow term \\
term \rightarrow digit\{print('digit')\} \\
term \rightarrow (expr)
\end{gathered}
$$

非终结符属性为val，终结符属性为$lexval$，则注释分析树有：

$9-5+2$：



$9-5*2$：



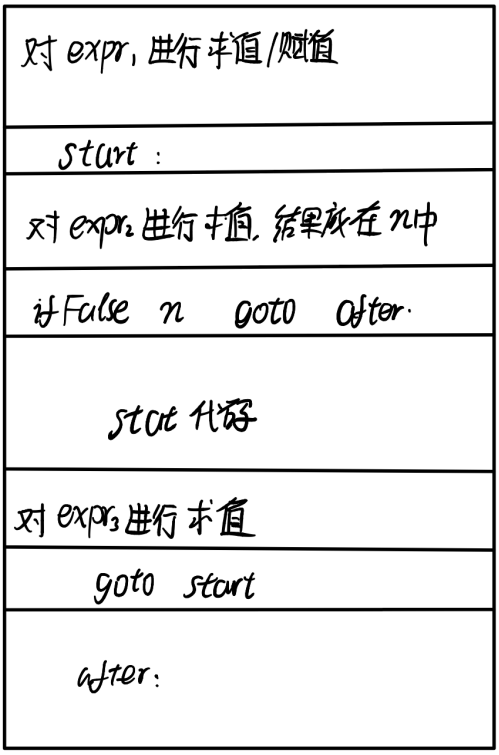
练习2.1.4：C语言和Java语言中的for语句具有如下形式：

for (expr1 ; expr2 ; expr3) stmt

第一个表达式在循环之前执行，它通常被用来初始化循环下标。第二个表达式是一个测试，它在循环的每次迭代之前执行。如果这个表达式的结果变成0，就退出循环。循环本身可以被看作语句{ stmt expr3 ;}。第三个表达式在每一次迭代的末尾执行，它通常用来使循环下标递增，故for语句的含义类似于

expr1 ; while(expr2) { stmt expr3 ;}

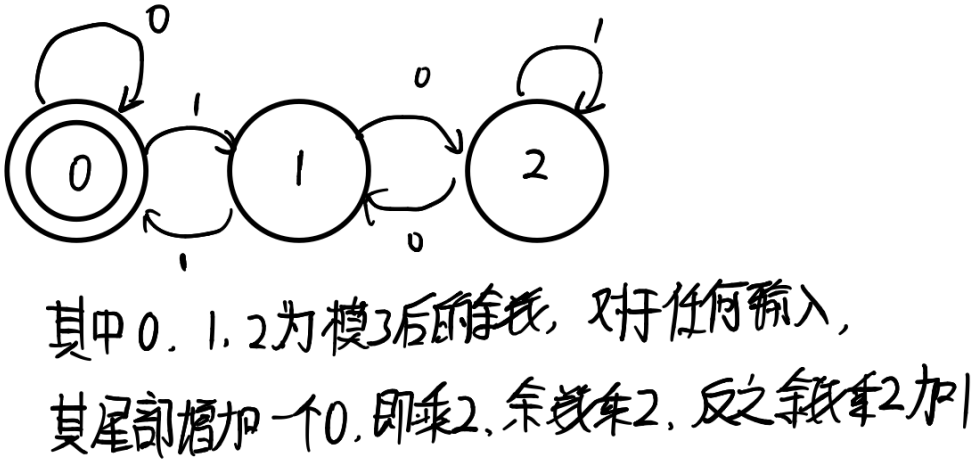
仿照图2-43中的类If，为for语句定义一个类For。




```
class For extends Stmt{
    Expr E1;
    Expr E2;
    Expr E3;
    Stmt S;
    public For(Expr expr1, Expr expr2, Expr expr3, Stmt stmt){
        E1 = expr1;
        E2 = expr2;
        E3 = expr3;
        S = stmt;
        Label start = new Lable();
        Lalel after = new Lable();
    }
    public void gen(){
        E1.gen();
        emit(start+":");
        Expr n = E2.rvalue();
        emit("ifFalse " + n.toString() + " goto " + after);
        S.gen();
        E3.gen();
        emit("goto " + start);
        emit(after + ":")
    }
}
```

练习2.1.5*：给出一个文法，生成所有被3整除的正整数的二进制串并给出证明。

首先给出DFA：



此DFA可以识别任何被3整除的2进制字符串，其后将之转换为同等计算能力的CFL，有

$$
\begin{aligned}
S &\to R_0 \\
R_0 &\to 1R_1|0R_0|\epsilon \\
R_1 &\to 1R_0|0R_2 \\
R_2 &\to 1R_2|0R_1
\end{aligned}
$$