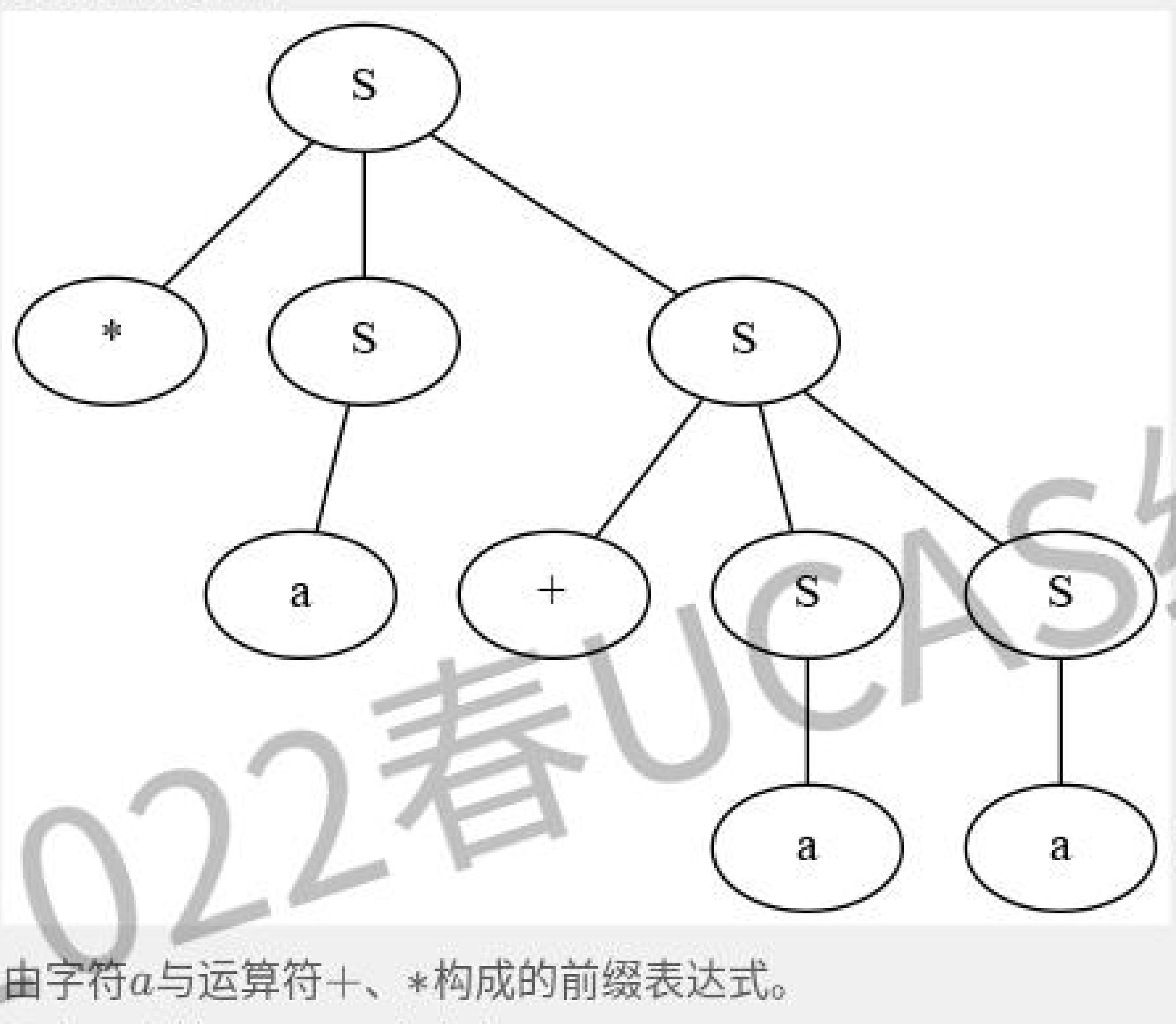
第二次作业

2.1.1. 考虑下面的上下文无关文法：

$$S \rightarrow +SS | * SS | a$$

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. （答案以最左推导为例，不唯一）
>
> $$S \Rightarrow *SS \Rightarrow *aS \Rightarrow *a + SS \Rightarrow *a + aS \Rightarrow *a + aa$$
>
> 2. 语法分析树如下：
>
> 
>
>
> 3. 由字符a与运算符$+$、$*$构成的前缀表达式。
> 4. 没有二义性。（证明言之有理即可）
> 对串的长度作归纳可以证明以下两个引理：对任意串s，若$S \Rightarrow^* s$则(1) s除本身外的任意前缀都有$*$、$+$的个数不小于a的个数；(2) s中$*$、$+$的个数比a的个数少1。
> 我们用反证法证明文法没有二义性：假设最短的有二义的串为s，显然$s \neq a$。无妨设s的第一个符号为$*$，则$s = *s_1s_2$且$S \Rightarrow^* s_1$，$S \Rightarrow^* s_2$。因为s_1、s_2短于s，所以s_1、s_2没有二义，由s有二义知存在另一种划分$*s_1's_2'$且$s_1 \neq s_1'$。这里不妨假设s_1'短于s_1，即s_1'是s_1的一个前缀，从而s_1'中$+$、$*$个数不少于a的个数，这与引理(2)矛盾，所以假设是错误的。

2.1.2. 考虑文法

$$num \rightarrow 101|1111|num\ 0|num\ num$$

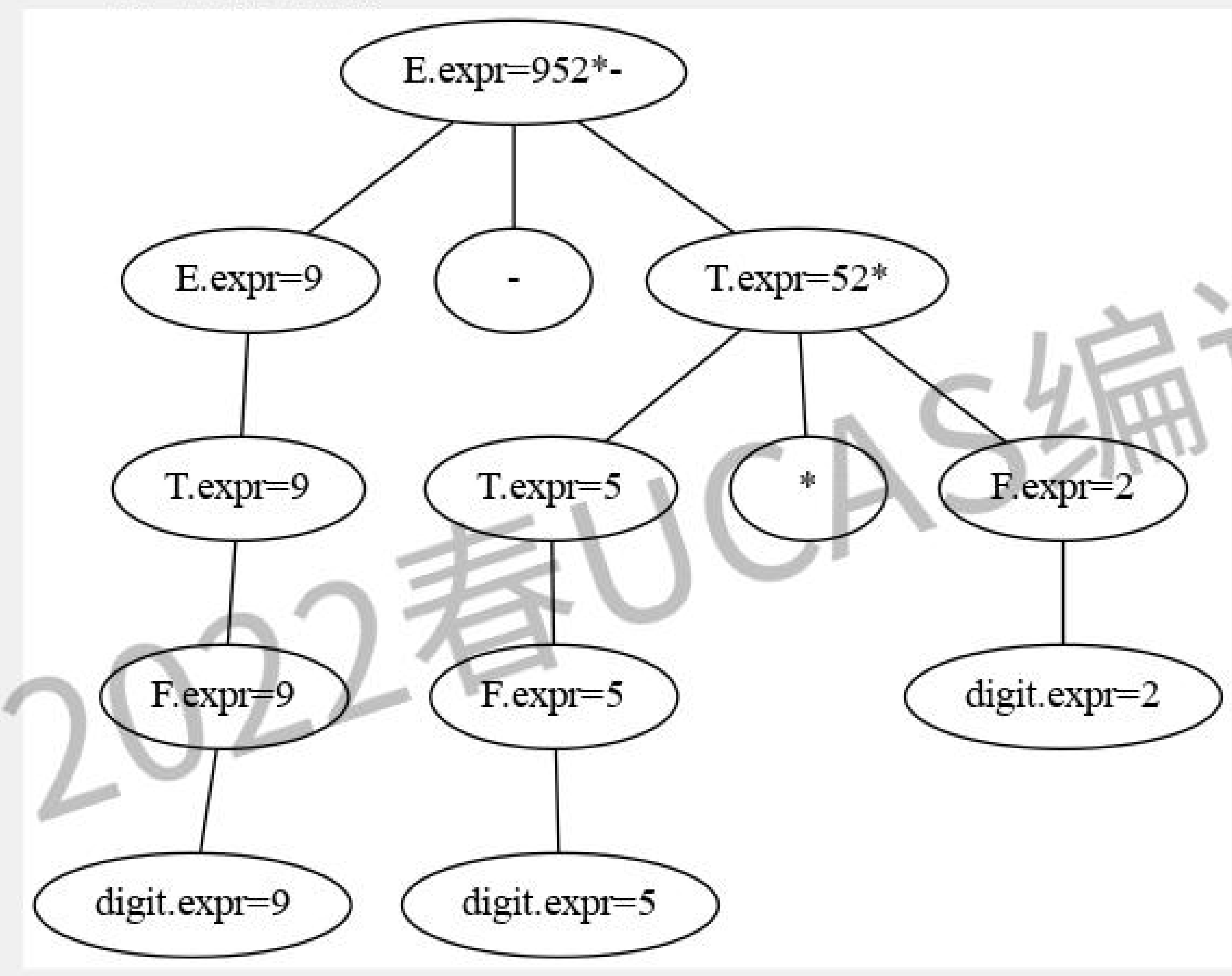
1. 证明：用下面文法生成的所有二进制串的值都能被 5 整除
（提示：对语法分析树的结点数目，即推导步数，使用数学归纳法）
1. 上面的文法是否能够生成所有能被 5 整除的二进制串？

> 1. （其他证明言之有理也可。）
> 使用数学归纳法，对该文法最终获得句子进行的推导步数（语法分析树节点）m进行归纳。
> 当 $m = 1$ 时，仅进行 1 次推导，仅有 $num \Rightarrow 101$ 和 $num \Rightarrow 1111$ 两种可能，两个句子分别表示数值 5 和 15，均能被 5 整除。设 $m \leq k (k \geq 1)$ 时，得到句子的数值都能被 5 整除，
>
> - 对于 $m = k + 1$：第 1 步推导必然为 $num \Rightarrow num_1\ 0$ 或 $num \Rightarrow num_1\ num_2$（下标仅为区分$num$的多次出现）。
> - 对于 $num \Rightarrow num_1\ 0$，设 num_1 由 k 步推导得到终结符号串 x，则 num 最终推导得到终结符号串 $x0$，由归纳假设知 x 可被 5 整除，故 $x0$ 必能被 5 整除；
> 对于 $num \Rightarrow num_1\ num_2$：$num_1$ 和 num_2 分别可经过不超过 k 步推导获得终结符号串 x_1、x_2。因此 num 经过 $k + 1$ 步推导获得终结符号串 x_1x_2，由归纳假设知 x_1、x_2 均可被 5 整除，故 x_1x_2 可被 5 整除。
> 综上，该文法获得的句子表示的数值都可被 5 整除。
>
> 2. 不能。
> 可举反例说明，例如该文法无法表示0（0）、25（11001）、35（100011）等。

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。

> 语法制导翻译方案

```
E -> E1 + T { print('+') }
        | E1 - T { print('-') }
        | T
T -> T1 * F { print('*') }
        | T1 / F { print('/') }
        | F
F -> digit { print(digit) }
        | ( E )
```

> 9-5+2的注释语法分析树
> 注意区分翻译语法分析树和注释语法分析树的概念（龙书第2版中文版P196页）
>
> 
>
>
> 9-5*2的注释语法分析树
>
> 
>

2.1.4. C 语言和 Java 语言中的 for 语句通常具有如下形式：

```
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`。

> 参考答案如下（不唯一）：

```
class For extends Stmt{
    Expr E1,E2,E3;
    Stmt S;
    Label start,after;
    public For(Expr x, Expr y, Expr z, Stmt s) {
        E1 = x;
        E2 = y;
        E3 = z;
        S = s;
        start = newlabel();
        after = newlabel();
    }
    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 + ":");
    }
}
```

2.1.5. 给出一个文法，生成所有被3整除的**正整数**的2进制串并给出证明。

> 分开考虑除以3余0、1、2的串即可。
>
> $$S \rightarrow S0|A1$$
> $$A \rightarrow S1|B0|1$$
> $$B \rightarrow A0|B1$$