2.2.1 考虑下面的上下文无关文法：

S -> S S + | S S \* | a

1. 试说明如何使用文法生成串aa+a\*
2. 试为该串构造语法分析树
3. 该文法生成的语言是什么？
4. 该文法具有二义性吗？

1）S -> S S\* -> S S + S\* -> a S + S\* -> a a + S\* -> a a + a\*

2） S

S S \*

S S + a

a a

3）该文法生成的语言是由a，+和\*构成的后缀表达式，即只包含加乘操作的逆波兰表达式

4）该文法没有二义性。证明：假设该文法有二义性，其中有最少推导步的二义性句子为α。

显然，α ！= a。

下不妨设α = α1α2\*

首先，推导的第一步必然是S -> S S \*，这样一来α1和α2的步数均少于α，故均无二义性，从而得到α也无二义性，这与假设矛盾。所以该文法没有二义性。

2.2.5

1）证明：用下面文法生成的所有二进制串的值都能被3整除

num -> 11 | 1001 | num 0 | num num

证明：11对应的十进制值为3，1001对应的十进制值为9

该文法所能生成的二进制串的形式为有任意多个11与任意多个1001组成，中间可以有0相隔。于是，能生成的二进制串的值为：

，其中i和j分别遍历11和1001在字符串中出现的位置。

显然该值被3整除。

2）上面的文法能否生成所有被3整除的字符串？

不能。考虑这样的串：10101，其二进制值为21，但是不能被该文法生成。

2.3.1

构建一个语法制导翻译方案，把算术表达式从中缀表示方式翻译为前缀表示方式。给出输入9-5+2和9-5\*2的注释分析树。

考虑到左结合和运算优先级，构建以下文法：

expr -> expr + term

| expr - term

| term

term -> term \* factor

| term / factor

| factor

factor -> digit | (expr)

随后给出从中缀表示到前缀表示方式的翻译方案：

expr -> {print("+")} expr + term

| {print("-")} expr - term

| term

term -> {print("\*")} term \* factor

| {print("/")} term / factor

| factor

factor -> digit {print(digit)}

| (expr)

9-5+2的注释分析树： + - 9 5 2

expr

{print("+")} expr + term

factor

{print("-")} expr - term 2 print(‘2’)

factor

term 5 print(‘5’)

factor

9 print(‘9’)

9-5\*2的注释分析树： - 9 \* 5 2

expr

{print("-")} expr - term

term {print("\*")} term \* factor

factor factor 2 print(‘2’)

9 print(‘9’) 5 print(‘5’)