文法的定义及语法树

定义

在形式语言理论中，语法[1]是一组*形式语言*字符串的*产生规则*[2]。规则描述了如何根据语言的语法从语言的字母表中形成有效的字符串。

形式语言的正式定义是由Noam Chomsky在1950年代首次提出的生成语法的经典形式化中，语法G由以下部分组成：

- **N**：*非终结符号的*有限集合，与由*G*组成的字符串不相交。
- **Σ**： 与*N*不相交的终结符的有限集合
- **P**：限集合*P*的*产生式规则*:

$$(\Sigma \cup N)^* N (\Sigma \cup N)^* -> (\Sigma \cup N)^*$$

- **S**:

总结起来，G就是一个如下的四元组

$$G = (N, \Sigma, P, S)$$

1956年，乔姆斯基首次形式化生成语法时，把这些语法分类成了现在的乔姆斯基体系。他们的区别在于越来越有严苛的产生规则。3-型语言到0-型语言。其中最重要的类型是类型2和类型3：上下文无关语法、常规语法。尽管这种语法比0型弱，但是他们的解析器可以得到有效的实现。如有限状态机。并且对于**上下文无关语法的有用子集**有已知算法生成的LL解析器和LR解析器识别生成相应语言

语法树

根据推导可以把一段语言字符串表示成为一个语法树(分析树)。

- 内部节点：非终结符
- 叶子节点：终结符
- 父子关系：推导式的左侧⊡推导式的右侧

不同的推导顺序会得到不同的语法树

如：

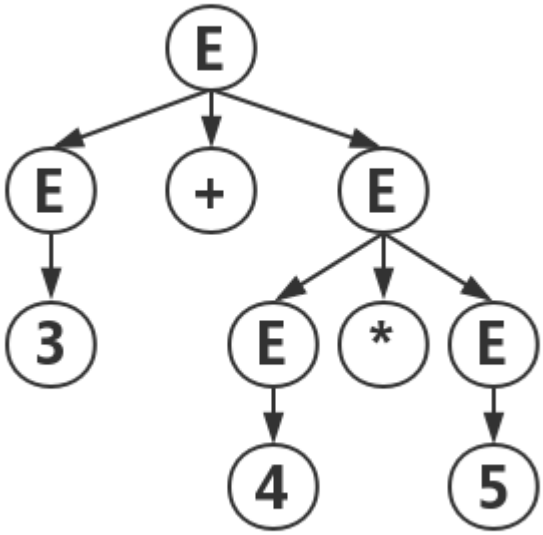
$$3 + 4 * 5$$

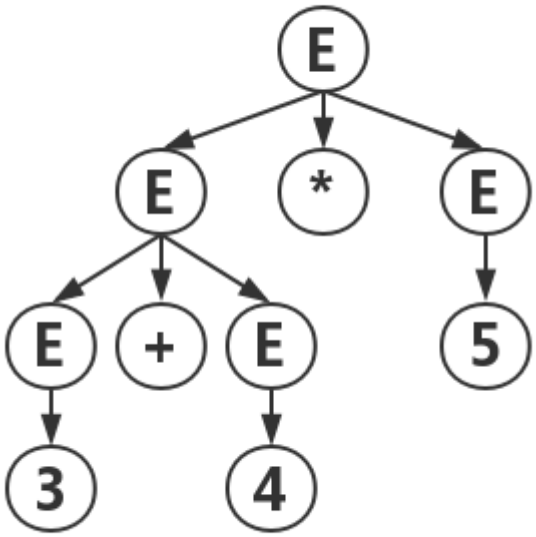
在一个

```
E -> num
    | id
    | E + E
    | E * E
```

这样的语法中所使用的推导式的顺序不同，会不同

这是因为这个语法中没有体现出来运算符的等级





这两个都是符合上述语法的语法树，而且遍历出子节点之后都是一样的，后序遍历之后的结果都是：

$$3+4*5$$

但是为了不产生起义，可以对源语法做出规则：

如

```
E -> T
    | E + T
T -> T * F
    | F
F -> num
    | id
```

这样这个语言就有了解析顺序。

---

1. 通常，在没有给定情况下，为了表述明确语法被称为形式语法↩

2. 在计算机科学领域中，产生规则是一个指定符号替换的重写规则，该规则可以递归执行以生成新的符号序列。↩