# 3 上下文无关文法

徐辉，xuh@fudan.edu.cn

本章学习目标：

- 掌握 LL(1) 文法。
- 掌握 LL(1) 解析表构造方法。

## 3.1 自顶向下解析

给定 CFG 文法和句子，找到由文法推导出该句子的过程称为解析。如何找到该推导方式呢？一种方式就是从根节点 `E` 开始，根据语法规则递归向下展开每个非终结符，直至最终生成的语法解释树与目标算式等价；如果出现不匹配的情况则立即回退。因此，该问题的难点是如何精准判断当前应采用哪个展开式，避免回退。一个基本思路是根据目标终结符决定当前应采用哪个生成式。为此，我们可以强制要求文法具备某些特性，如 LL（1）(Left-to-right, Left most, lookahead 1 symbol)。

## 3.2 LL(1) 文法

LL(1) 文法有两个基本要求，一是不含左递归，二是无回溯特性。下面对这两个特性进行探讨。

### 3.2.1 左递归和消除

对一条文法规则来说，如果其右侧推导出的第一个符号与左侧符号相同，则存在左递归问题，如（`E` ↦ `E OP1 E1`)。左递归可能会使搜索过程无限递归下去，无法终止。一般可以采用下列方式对左递归文法进行修改。

$$
\begin{aligned}
&\texttt{X} \mapsto \texttt{X <a> | X <b> | <c> | <d>} \\
&\qquad\quad \Downarrow \\
&\texttt{X} \mapsto \texttt{<c> Y | <d> Y} \\
&\texttt{Y} \mapsto \texttt{<a> Y | <b> Y |} \ \epsilon
\end{aligned}
\tag{3.1}
$$

将该方法应用于图 **??**中的文法，可消除其左递归问题。结果如图 3.1所示。

### 3.2.2 无回溯语法

对于每个非终结符的任意两个生成式，如果其产生的首个终结符均不同，则前瞻一个单词总能够选择正确的规则。当生成式的首个字符是非终结符时，应对该非终结符递归展开直至遇到终结符为止。

$$
\begin{aligned}
&X \xrightarrow{[i]} < a > ... \\
&X \xrightarrow{[j]} < b > ... \\
&X \xrightarrow{[k]} Y ... \xrightarrow{[l]} < c > ... \\
&...
\end{aligned}
\tag{3.2}
$$

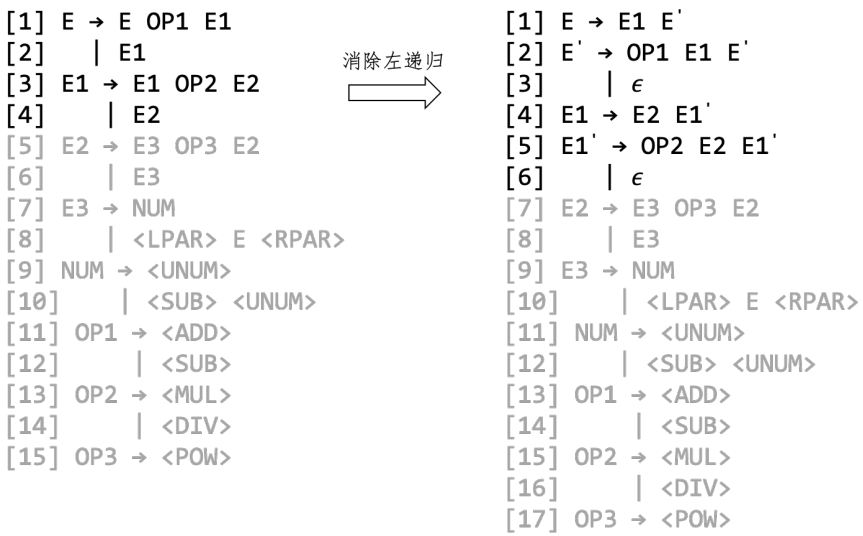


图 3.1: 消除左递归后的 CFG 语法

当文法规则存在回溯问题时，可以通过提取左公因子消除回溯。

$$\begin{aligned} & X \rightarrow < a > A | < a > B | < b > \\ & \qquad \Downarrow \\ & X \rightarrow < a > Y | < b > \\ & Y \rightarrow A | B \end{aligned} \tag{3.3}$$

## 3.3 构造 LL(1) 解析表

我们定义 $First(X \xrightarrow{[i]} \beta_1\beta_2...\beta_n)$ 表示 X 的第 i 条产生式的首字符集合。如果 $\epsilon \notin \beta_1$，则 $First(X) = First(\beta_1)$；如果 $\epsilon \in \beta_1 \& ... \& \epsilon \in \beta_i$，则 $First(X) = First(\beta_{i+1})$。如果 $\epsilon \in \beta_1 \& ... \& \epsilon \in \beta_n$，我们还需考虑 $X$ 之后可能出现的字符 $Follow(X \xrightarrow{[i]} ...)$，并据此决定是否采用该 $X \mapsto \epsilon$ 的生成式。因此我们使用 $First^+(X \xrightarrow{[i]} \beta)$X 的第 i 条产生式的首字符集合（不含 $\epsilon$）。

$$First^+(X \mapsto \beta) = \begin{cases} First(\beta), & \text{if } \epsilon \in \beta \\ First(\beta) \cup Follow(A), & \text{otherwise} \end{cases}$$

基于上述定义，我们可以准确描述出无回溯语法的必要性质。

$$\forall 1 \le i, j \le n, First^+(A \rightarrow \beta_i) \cap \text{ First }^+(A \rightarrow \beta_j) = \emptyset$$

表 3.1展示了图 3.1中文法规则的 $First$ 集合；其每一行表示一个非终结符，每一列表示一个终结符，单元格内容表示对应的规则编号。

表 3.1: 记录每条生成式的 $First$ 集合。

| | <UNUM> | <ADD> | <SUB> | <MUL> | <DIV> | <POW> | <LPAR> | <RPAR> | $\epsilon$ |
|---|---|---|---|---|---|---|---|---|---|
| E | [1] | | [1] | | | | [1] | | |
| E' | | [2] | [2] | | | | | | [3] |
| E1 | [4] | | [4] | | | | [4] | | |
| E1' | | | | [5] | [5] | | | | [6] |
| E2 | [7] | | [7] | | | | [7] | | |
| E2' | | | | | | [8] | | | [9] |
| E3 | [10] | | [10] | | | | [11] | | |
| NUM | [12] | | [13] | | | | | | |
| OP1 | | [14] | [15] | | | | | | |
| OP2 | | | | [16] | [17] | | | | |
| OP3 | | | | | | [18] | | | |

进一步消除表 3.1中的 $\epsilon$ 字符便可以得到 $First^+$ 或 LL(1) 解析表 3.2。基于无回溯文法的特性，该表的所有单元格至多存在一条规则。通过查表便可以实现精准快速解析。

表 3.2: LL(1) 解析表：生成式的 $First^+$ 集合。

| | <UNUM> | <ADD> | <SUB> | <MUL> | <DIV> | <POW> | <LPAR> | <RPAR> |
|---|---|---|---|---|---|---|---|---|
| E | [1] | | [1] | | | | [1] | |
| E' | | [2] | [2] | | | | | [3] |
| E1 | [4] | | [4] | | | | [4] | |
| E1' | | [6] | [6] | [5] | [5] | | | [6] |
| E2 | [7] | | [7] | | | | [7] | |
| E2' | | [9] | [9] | [9] | [9] | [8] | | [9] |
| E3 | [10] | | [10] | | | | [11] | |
| NUM | [12] | | [13] | | | | | |
| OP1 | | [14] | [15] | | | | | |
| OP2 | | | | [16] | [17] | | | |
| OP3 | | | | | | [18] | | |

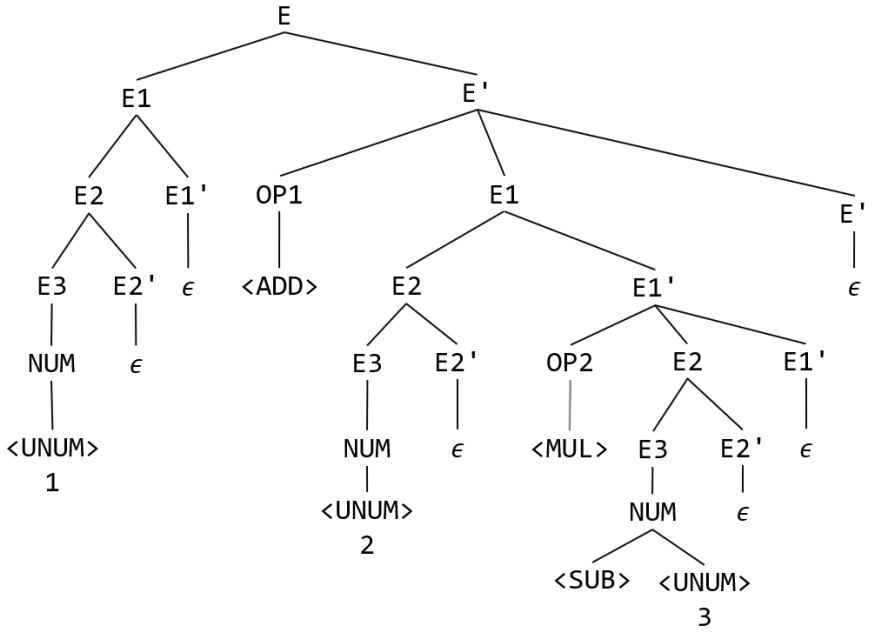


图 3.2: 使用 LL(1) 语法解析 1+2*-3 得到的语法解析树

基于 LL(1) 文法解析算式 1+2*-3 得到的语法解析树（图 3.2）。

## 3.4 练习

上节课练习编写的正则表达式解析文法是否是 LL(1) 文法？如果不是将其改写为 LL(1) 文法；为该 LL(1) 文法构造解析表。