7. LL 分析器

我们之前实现的 TEST 语言的语法分析器是基于递归的自上而下分析器，由于递归分析的时空成本较高，本章我们采用非递归的自上而下分析器，以克服时空成本高的问题。非递归的自上而下分析器就是 LL 分析器，它首先构造一个分析决策表，然后根据当前情况查决策表并做出决策，逐步分析到生成整棵语法树。

7.1 构造 LL 分析决策表

7.1.1 首先计算 FIRST 集合

假定是β是文法 G 的任意符号串，或$\beta \in (V_t \cup V_n)^*$，则

FIRST(β)={ $a \mid \beta \Longrightarrow^* a \ldots, a \in V_t$}。若$\beta \Longrightarrow^* \varepsilon$，则规定$\varepsilon$ ∈FIRST(β)。

FIRST 集合构造方法如下：

对于文法中的符号 X∈ $V_t \cup V_n$，其 FIRST(X)集合可反复应用下列规则计算，直到其 FIRST(X)集合不再增大为止。

(1) 若 X∈ V_t，则 FIRST(X)={X}。

(2) 若 X∈ V_n，且具有形如 X→aα的产生式(a∈ V_t)，或具有 X→ε的产生式，则把 a 或ε加进 FIRST(X)。

(3) 设 G 中有形如 X→$Y_1Y_2\cdots Y_k$的产生式，若 Y_1∈ V_n，则把 FIRST(Y_1)中的一切非ε符号加进 FIRST(X)；对于一切 2≤ $i \le k$，若 $Y_1Y_2\cdots Y_{i-1}$均为非终结符号，且ε ∈FIRST(Y_j)，1≤ $j \le i-1$，则将 FIRST(Y_j)中的一切非ε符号加进

FIRST(X)；但若对一切 1≤ $i \le k$，均有ε ∈FIRST(Y_i)，则将ε加进FIRST(X)。

7.1.2 计算 FOLLOW 集合

假定 S 是文法的开始符号，对于 G 的任何非终结符号A，则

FOLLOW(A)={ a | S⟹* …$A\ a$ …, $a \in V_t$}

若 S⟹* …A，则规定$∈FOLLOW($A$)，$是句尾标志。

FOLLOW(A)就是在所有句型中紧接A后出现的终结符或$。对于文法符号$A \in V_n$，FOLLOW($A$)集合的计算可反复应用下列规则，直到 FOLLOW(A)集合不再增大为止：

(1) 对于 S，令$∈FOLLOW(S)。

(2) 若 G 中形如 B→$\alpha A\beta$的产生式，且$\beta \neq \varepsilon$，则将 FIRST(β)中的一切非ε符号加进 FOLLOW(A)中。

(3) 若 G 中有形如 B→αA或 B→$\alpha A\beta$的产生式，ε ∈FIRST(β)，则FOLLOW(B)中的全部元素均属于 FOLLOW(A)。

7.1.3 构造分析决策表

(1) 对文法的每个产生式 $A \to \alpha$ ，执行(2)和(3)

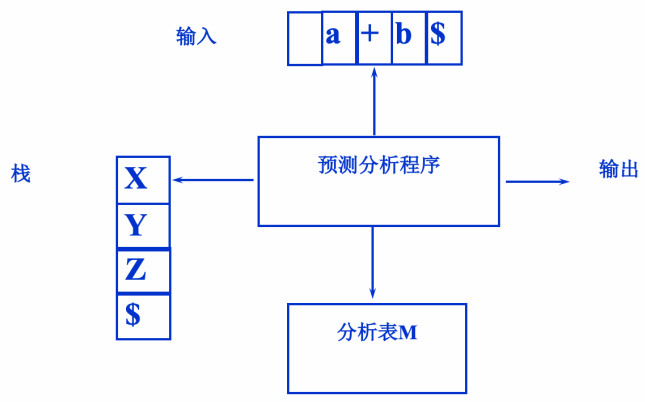
(2) 对 FIRST(α)的每个终结符 a，把 $A \to \alpha$ 加入 M[A, a]

(3) 如果ε在 FIRST(α)中，对 FOLLOW(A)的每个终结符 b（包括$），把 $A \to \alpha$加入 M[A, b]

(4) M 中其它没有定义的条目都是 error

7.2 LL 分析过程

LL 分析器结构如下图：



需要维护一个栈，一个输入序列，预测分析程序会根据分析表 M 做出决策，最后生成整棵语法树。

预测分析程序：

初始化： S $在栈里，其中 S 是文法的开始符号并且在栈顶； w $ 为输入序列

让 ip 指向 w $ 的第一个符号

主程序：令 X 等于栈顶符号，并且 a 等于 ip 指向的符号；

Repeat

If X 是终结符

If X == a

把 X 从栈顶弹出并推进 ip；

Else Error();

Else if M[X,a] = X → $Y_1Y_2\cdots Y_k$ /*X 是非终结符*/

从栈中弹出 X；

把 $Y_k, Y_{k-1}, \cdots Y_1$ 依次压入栈，Y_1 在栈顶；

输出产生式 X → $Y_1Y_2\cdots Y_k$ /*对应子树 X 为父节点，$Y_1Y_2\cdots Y_k$ 为子节点*/

Else Error();

Until X == $且 ip 指向$ /*栈空且输入序列到尾部*/

7.3 作业

1. 给定语法： $E \rightarrow TE'$

$E' \rightarrow +\ TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *\ FT' \mid \varepsilon$

$F \rightarrow (E) \mid \mathbf{id}$

实现一个 LL 分析器，可以对任何加乘运算序列（如 id + id * (id + id) ）输出其语法树（要可视化）。

2. 给定上次作业的语法，即实现基于递归的自上而下分析器的语法，本次作业实现一个 LL 分析器，可以对符合该语法的输入程序进行语法分析，假设该程序符合语法，则输出语法分析成功，如果该程序不符合语法，则输出相应的语法错误的行和行号。