编译原理实验指导书

Compilers Principles Experiment Instruction Book

陈贺敏　王林

燕山大学软件工程系

实验 2 自顶向下的语法分析程序

2.1 实验目的

（1）熟练掌握 LL(1)分析表的构造方法。

（2）掌握设计、编制和调试典型的语法分析程序，进一步掌握常用的语法分析方法。

2.2 实验任务

根据 LL(1)分析法自己编写一个语法分析程序，语言不限，文法不限。

2.3 实验内容

2.3.1 实验要求

你的程序应具有通用性，能够识别由词法分析得出的词法单元序列是否是给定文法的正确句子（程序），并能够输出分析过程和识别结果。

2.3.2 输入格式

1、一个包含源代码的文本文件，此时需要和词法分析程序进行对接，通过一次扫描同时完成词法分析和语法分析；

2、通过实验一中借助 Flex 工具或自己编写的词法分析程序，得到相应的词法单元序列，然后将其作为此次语法分析程序的输入。

注：两种输入格式，二选其一即可。

2.3.3 输出格式

实验二要求通过标准输出打印程序的运行结果（包括 First 集、Follow 集、LL(1)分析表），此外，要求可以保存 LL(1)分析表。

你的程序需要输出语法分析过程和相应的分析结果（即此串是否为 LL(1)文法的句子）。

2.3.4 提交要求

1) 可以是 Java、C、Python 等任意语言编写的可被正确编译运行的源程序。

2）一份实验报告，内容包括：实验目的、实验任务、实验内容，算法描述，程序结构，主要变量说明，程序清单，调试情况及各种情况运行结果截图，心得体会。

2.3.5 样例

此部分输出样例可参考课本例题中对输入串的分析过程表（教材 P94-95 的表 4.4 和表 4.5）。

2.4 实验指导

词法分析的下一阶段是语法分析，语法分析是编译原理的核心部分，其在编译器模型中的位置如图 2-1 所示。语法分析的作用是识别从词法分析器获得的一个由词法单元序列组成的串是否是给定文法的正确句子。目前语法分析常用的方法有自顶向下分析和自底向上分析

两大类。此次实验采用自顶向下的 LL(1)分析方法。

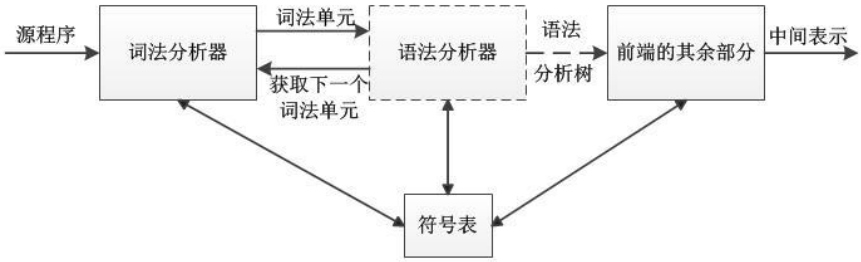

图 2-1 编译器模型中语法分析器的位置

2.4.1 LL(1)分析法概述

所谓 LL(1)分析法，就是指从左到右扫描输入串（源程序），同时采用最左推导，且对每次直接推导只需向前看一个输入符号，便可确定当前所应当选择的规则。实现 LL(1)分析的程序又称为 LL(1)分析程序或 LL(1)分析器。

一个文法要能进行 LL(1)分析，那么这个文法应该满足：无二义性，无左递归，无左公因子。LL(1)的语法分析程序包含了三个部分，总控程序，预测分析表函数，先进先出的语法分析栈。

2.4.2 预测分析程序

LL(1)预测分析程序的总控程序在任何时候都是按 STACK 栈顶符号 X 和当前输入符号 a 来决定做何种操作的。其具体工作过程可用示意图 2-2 表示。其中：“#”句子括号即输入串的括号；“S”文法的开始符号；“X”存放当前栈顶符号的工作单元；“a”存放当前输入符号 a 的工作单元。

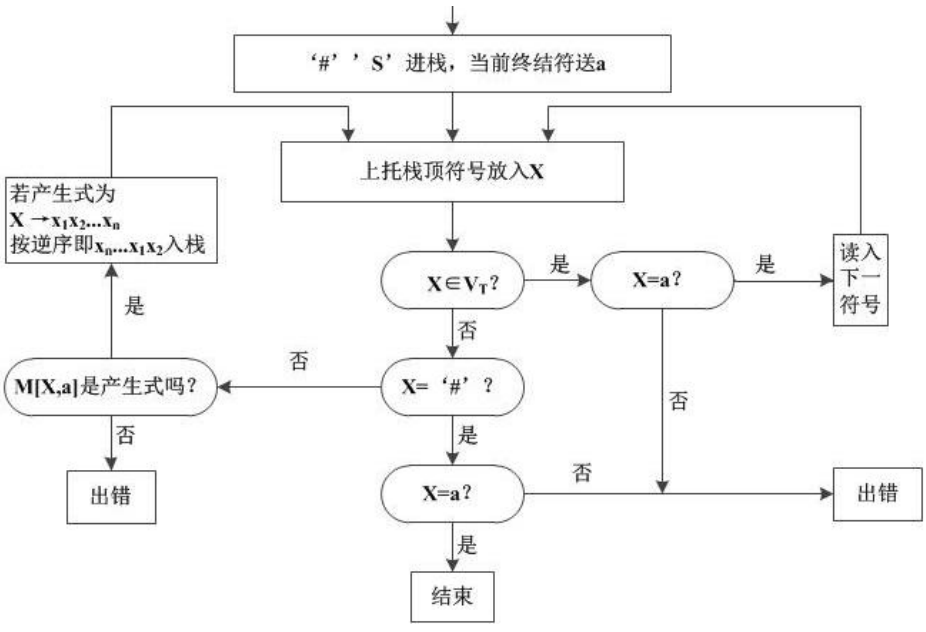

图 2-2 预测分析程序的框图

从图 2-2 可以看出，对于任何（X，a），总控程序每次都执行下述三种可能的动作之一：

1）若 X = a = ‘#’，则宣布分析成功，停止分析过程。

2）若 X = a≠ #’，则把 X 从 STACK 栈顶弹出，让 a 指向下一个输入符号。

3）若 X 是一个非终结符，则查看预测分析表 M。若 M[A，a]中存放着关于 X 的一个产生式，那么，首先把 X 弹出 STACK 栈顶，然后，把产生式的右部符号串按反序一一弹出 STACK 栈（若右部符号为ε，则不推什么东西进 STACK 栈）。若 M[A，a]中存放着“出错标志”，则调用出错诊断程序 ERROR。

2.4.3 构造 LL(1)分析表

对于此次语法分析程序中用到的分析表，可根据自己实际情况，选择以下一项实现分析算法中分析表的构造：

1）直接输入根据已知文法构造的分析表 M;

2）输入文法的 FIRST(α)和 FOLLOW(U)集合，由程序自动生成文法的分析表 M；

3）输入已知文法，由程序自动构造文法的分析表 M。