编译器构造实验

Lab5—实验 4

手工编写递归下降预测分析程序

姓名：郝裕玮

班级：计科 1 班

学号：18329015

目录

1 设计 Oberon-0 语言的翻译模式

由于学艺不精，期末复习进度较紧，本问未能实现。在这里向助教和老师致以深深的歉意！非常抱歉！

2 编写递归下降预测分析程序

由于学艺不精，期末复习进度较紧，本问未能实现。在这里向助教和老师致以深深的歉意！非常抱歉！

3 语法分析讨论：自顶向下 vs. 自底向上

（1）分析技术的简单性，包括分析程序是否易于调试：

对于自顶向下，主要文法为 LL(1)文法，它只需要扫描 lookahead 然后就可以继续执行相关操作。

对于自底向上，主要文法为 LR(0)，SLR(1)，LR(1)，LALR(1)，它需要栈来保存信息。

所以显然自顶向下的技术相较于自底向上更为简单，其分析程序也更易于调试。

（2）分析技术的通用性，即能处理的语言范围：

对于自顶向下，主要文法为 LL(1)文法，文法为继承属性。

对于自底向上，主要文法为 LR(0)，SLR(1)，LR(1)，LALR(1)，文法为综合属性和继承属性。

再结合 LR(0)，SLR(1)，LR(1)，LALR(1)和 LL(1)对冲突的处理，显然自底向上能处理的语言范围更大。

（3）是否便于表达语义动作以完成语法制导翻译：

对于自顶向下，它必须每一次都把继承属性以参数形式传入下一层函数中。所以它必须考虑参数与继承属性的对应以及参数个数。

对于自底向上，则是用函数的返回值来返回综合属性。而由于函数返回值只有一个，所以自底向上可生成一个对象来返回所有的属性信息。

（4）是否易于实现出错恢复：

对于自顶向下，若发现错误，考虑到错误恐慌，还需要忽略并弹出一部分正在返回的 token。

对于自底向上，若发现错误，只需弹出当前的 token，然后即可继续对程序进行分析。

综上所述，自底向上更易于实现出错恢复。

（5）若以表格驱动方式取代递归程序实现，则分析表大小的优劣如何：

递归下降可以使用传入，返回参数的方式传递属性；而表驱动需要入栈出栈比较复杂。

但是递归下降有个先天问题就是调用层级比较深，而调用层级深可能会导致性能下降以及栈溢出。如果用显示栈就能避免这个问题。

（6）分析速度：

对于自顶向下，只需要一个 token 即可开始递归或匹配。

对于自底向上，需要扫描入栈多个 token 才能决定移进(shift)或归约(reduce)。

所以显然自顶向下比自底向上速度更快。

4 实验心得

实验 4 通过对自顶向下和自底向上的多项对比分析让我对这两种语法有了更深层次的理解。但由于时间不足，学艺不精等多方面因素导致未能完成全部环节。在这里再次向助教和老师表示抱歉！