编译原理实验二报告

--居于 LL1 文法的语法分析

南京大学 软件学院 韦祖策 131250148

1. 目标

本次实验的目的是对编译器进行词法分析的过程进行模拟，我选择了在实际中更为通用的自底向上的词法分析器的分析过程，最终产生规约序列。对于 LR(0)和 LR(1)问题，我的程序对于 LR(0)和 LR(1)是通用的，因为只要给出合法的 parsing table 和上下文无关文法， 程序就能进行相应的词法分析，而 parsing table 和文法都是用户输入文件给出。

2. 内容概述

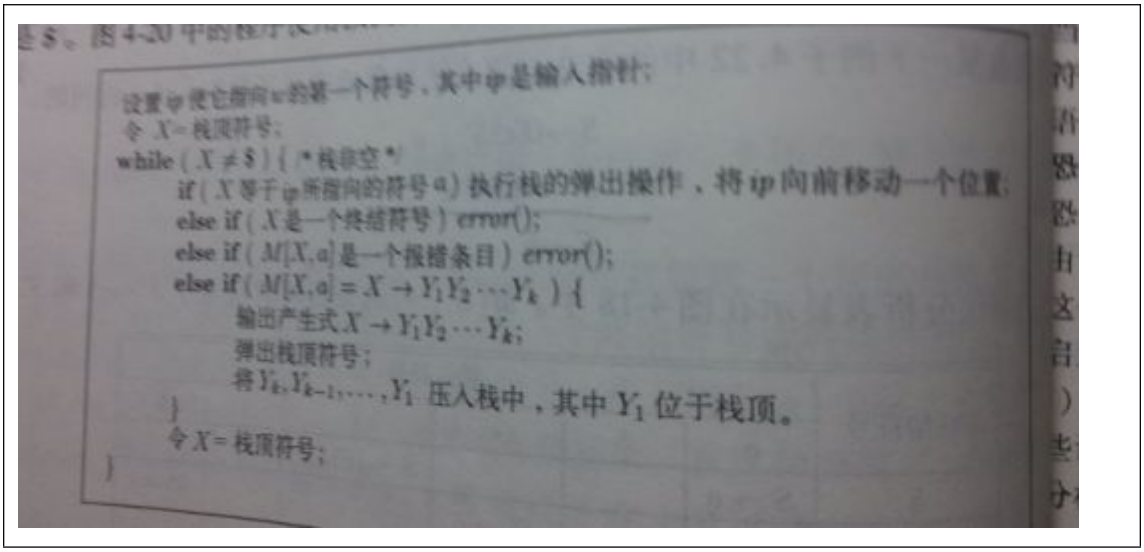
本文档描述了编译原理课程实验中，语法分析器部分的实验内容，实验方案以及结果。

3.实验环境

操作系统：win8.1
编译器：eclipse
使用的工具：github
编码格式：utf-8

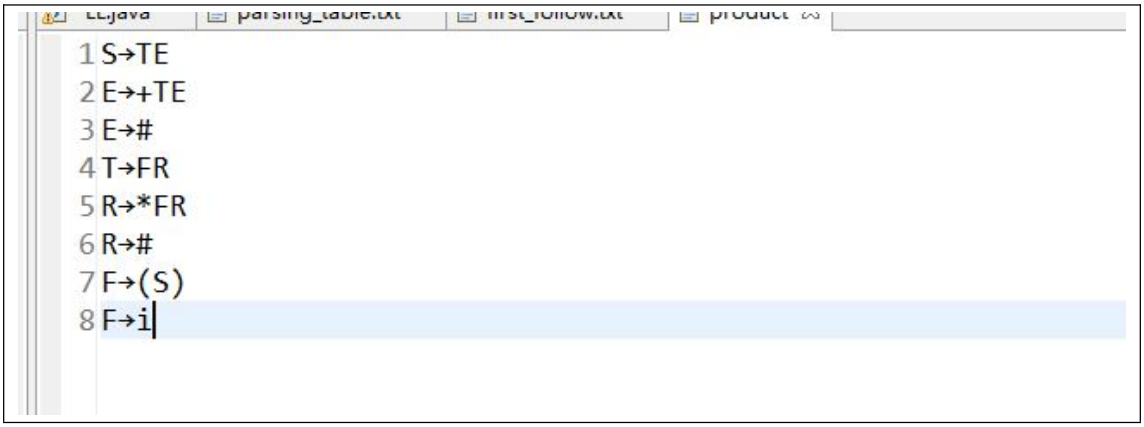
4.思路和核心思想

根据给出的文法，文法要求是非二义性的、非左递归的上下文无关文法，输入到 product 的文件中。从文件中读出输入的文法，先通过对输入的文法求每一个产生式中的非终结符的 first 和 follow 集合。来构建 LL(1)的预测分析表。然后使用预测分析表来进行表格驱动，对于输入的串进行预测分析，使用书上的算法，现在在此处附上这算法，因为比较难打字，我附图：



5.测试输入与输出：

测试输入的文法产生式是书中的例子文法 4-28，因为表示空的字符我打不出，就使用了#代替。书中的产生式还中的左边可以有或（|）进行连接，在我的输入中只能分开，当作多个产生式，同时不能有其他字符。对于终结符，只能有一个字母限定，书中的 id 我使用 i 代替。由于输入的原因，我把把书中的输入形式做了一下转变，实质上是同一个文法。
同时，输入串中
输入的文件在项目当中，截图如下：



程序中产生的 first 和 follow，如下，也存在项目中的文件 first_follow.txt 中：

```
first集合:
first(S)={ ( , i}
first(E)={ + , #}
first(T)={ ( , i}
first(R)={ * , #}
first(F)={ ( , i}
follow集合:
fl(S)={ #,)}
fl(E)={ #,)}
fl(T)={ +,#,)}
fl(R)={ +,#,)}
fl(F)={ *,+,#,)}
```

产生的预测分析表：

	+	$	*	(	)	i
S				→TE		→TE
E	→+TE	→#			→#	
T				→FR		→FR
R	→#	→#	→*FR		→#	
F				→(S)		→i

Done

预测分析过程：

步骤	分析栈	剩余输入串	所用产生式或匹配
1	S#	i+i*i#	S→TE
2	TE#	i+i*i#	T→FR
3	FRE#	i+i*i#	F→i
4	iRE$	i+i*i$	i匹配
5	RE#	+i*i#	R→#
6	E#	+i*i#	E→+TE
7	+TE$	+i*i$	+匹配
8	TE#	i*i#	T→FR
9	FRE#	i*i#	F→i
10	iRE$	i*i$	i匹配
11	RE#	*i#	R→*FR
12	*FRE$	*i$	*匹配
13	FRE#	i#	F→i
14	iRE$	i$	i匹配
15	RE#	#	R→#
16	E#	#	E→#
1	#	#	分析成功

这些测试的结果都会以文件的形式产生在项目里的文件中。也会打印到控制台上。

6.我的感受

反正一开始是感觉做什么都不知道，然后还去网上找别人写的来看，写这个花的时间也比较多，临近考试周了，写出了来了也是一个成就感很强的，对之前不懂的地方又回去理解一番。还是比较有意义的。