Lab1: Transition Table Based Lexical Analyzer

唐龙炜

2014 年 4 月 9 日

1 概述

At the first phase of a compiler,the main task of the lexical analyer is to read the input characters of the source program,group them into lexemes.A token is a pair consisting of a token name and an optional attribute.The token name is an abstract symbol representing a kind of lexical unit. and in my implementation,there are seven kind of token name as identifier,keywords,number constant,arithmetic operator,assign symbol(:=) and spacing,comparison.as the table list blow:

number	记号名	描述	例子
1	keywords	PASCAL 语言的关键字	var begin else …
2	number constant	数字常量	123,343 …
3	identifier	标识符	i j abc …
4	arithmetic operator	数学运算符	+-*/
5	assign	赋值符号	：=
6	comparison	比较符号	>= ,>,<,=<,…
7	spacing	源程序中的空格	换行, 连续空格

Every recognized token will have the following format:

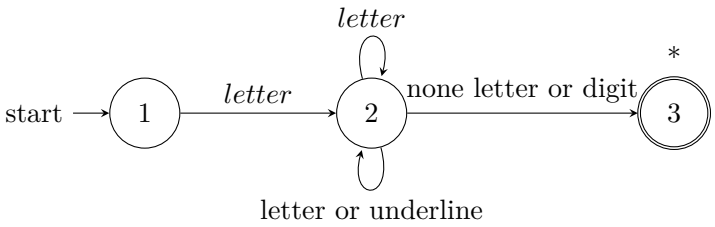
```
<token name,token attribute>
```

2 转换图

将每一种记号对应的模式转换为状态转换表, 从状态转换表写出程序相对来讲比较容易，以下是上述记号的状态转换图（dfa）及其对应的状态转换表：
注意：

- 没有关键字的状态转换表，因为关键字可以当成是标识符的一个子集，因此先识别出标识符 *Token*，在通过 *Token* 的相关属性判断是否是关键字。
- 在数字常量的识别中，并没有考虑小数、指数、负数，只考虑了简单的无符号数如 123，245 …

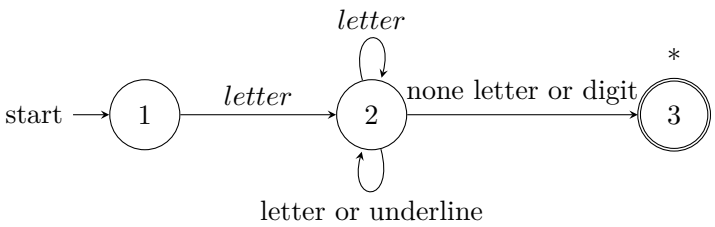
1. keywords
2. number constant



由此得到的状态转换表：

state\input	digit	other
1	2	-1
2	2	3

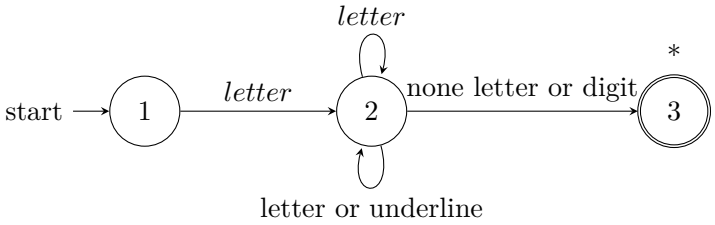
3. identifier



由此得到的状态转换表：

state\input	a-zA-Z	underline	digit	other
1	2	-1	-1	-1
2	2	2	2	3

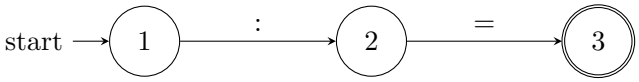
4. arithmetic operator



由此得到状态转换表：

state\input	a-zA-Z	underline	digit	other
1	2	-1	-1	-1
2	2	2	2	3

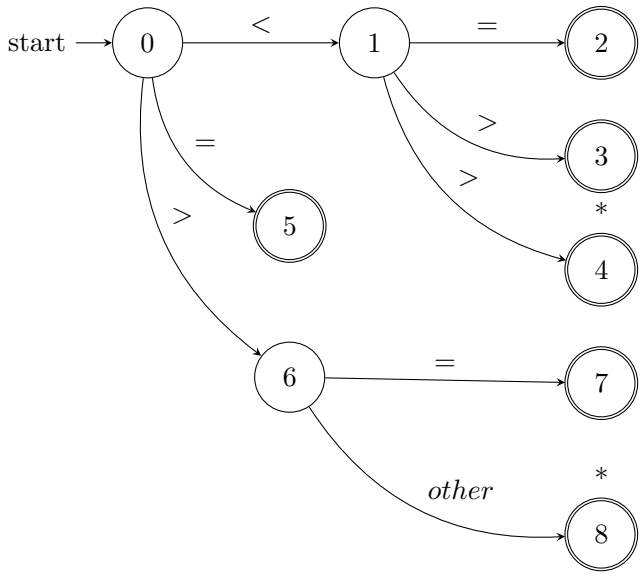
5. assign



由此得到状态转换表：

state\input	:	=
1	2	-1
2	-1	3

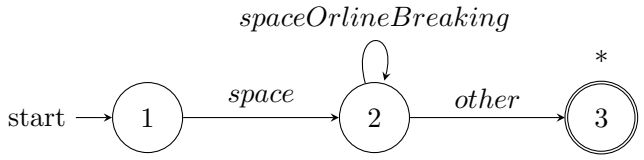
6. comparison



由此得到状态转换表：

state\input	>	=	<	other
0	6	5	1	-1
1	3	2	-1	4
6	-1	7	-1	8

7. spacing



由此得到状态转换表：

state\input	spaceOrlineBreaking	other
1	2	-1
2	2	3

3 实现

在实现中我,, 编译平台为 gcc 4.4.7,os:centos,ide:codeblocks, 为每一个记号的模式设计了一个 move 函数用于模拟状态转换表的行为,get 函数利用 move 函数实现具体的模式的分析, 详见代码注释. 在 main 函数中，对于各个模式的识别顺序有并没有严格的要求, 因为仅从首字母便可以分析出所属的模式。程序就收 source 文件的输入，并将记号流输出到一个 out 文件, 具体格式见文件目录。