第三章 词法分析 →∪Σε

3.8 词法分析器生成工具的设计

有穷自动机（NFA或DFA）是识别器，它们只能对每个可能的输入串简单地回答“是”或“否”。

正则表达式: a\*b+ --- > NFA 或者DFA

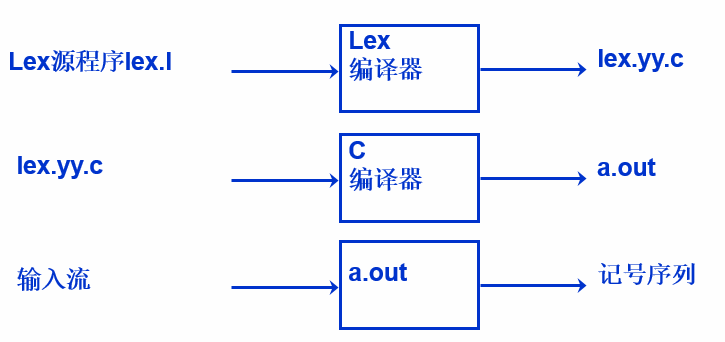
输入串: aaaabb

aaaabba

词法分析器（主要任务）: 读入源程序的输入字符、将它们组成词素，生成并输出一个词法单元序列，每个词法单元对应于一个词素。

Lex词法分析器 (第3.5节)

用Lex建立建立词法分析器的步骤



Lex程序包括三个部分

声明部分

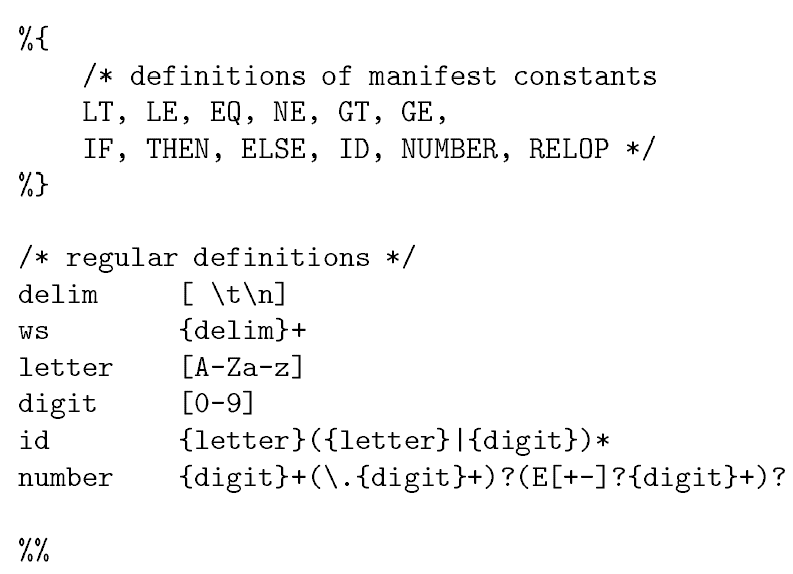
%%

翻译规则 (转换规则)

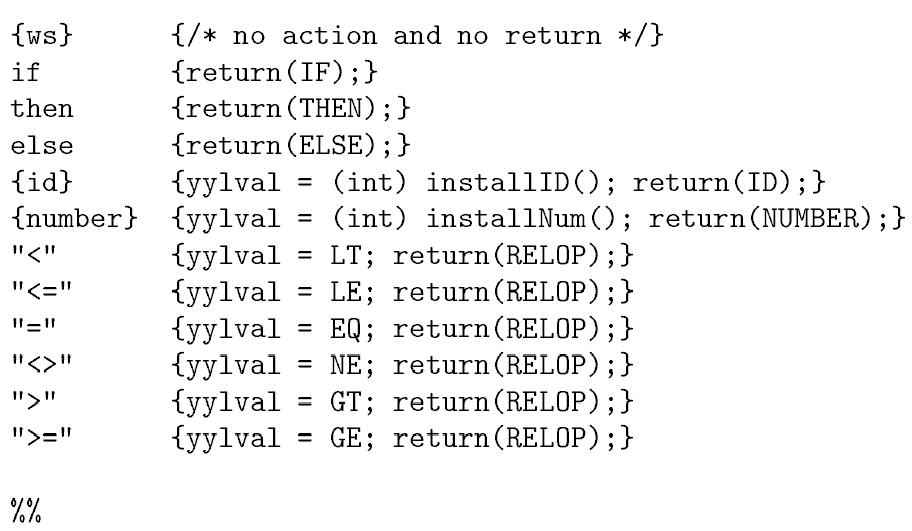
%%

辅助过程

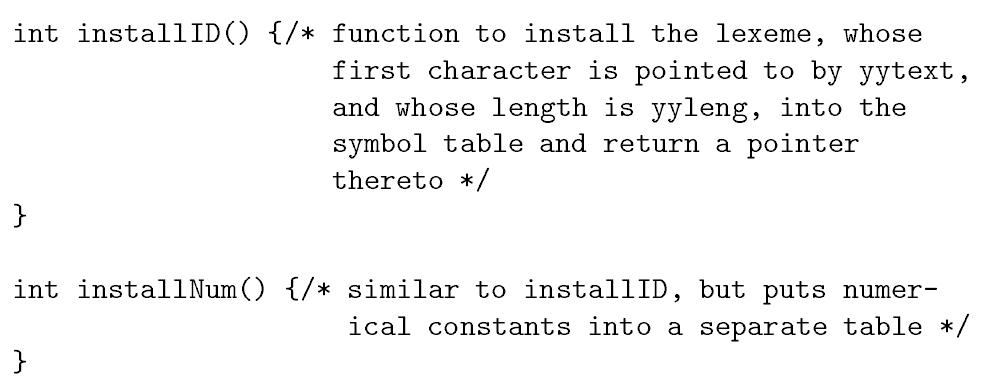
* 声明部分: 变量和明示常量的声明，以及正则定义



* 翻译规则: 模式 {动作}, 其中每个模式是一个正则表达式, 动作部分是代码片段



* 辅助函数: 各个动作需要使用的所有辅助函数。



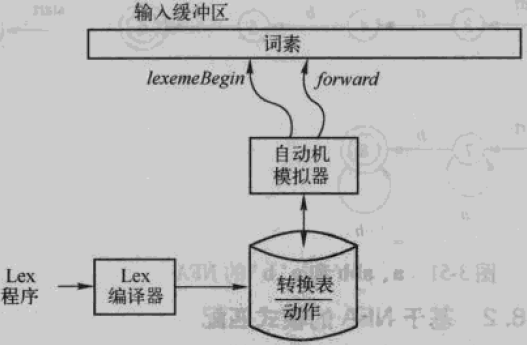
当输入的多个前缀与一个或多个模式匹配时，Lex用如下规则选择正确的词素:

1. 总是选择最长的前缀

abcefg

1. 如果最长的可能前缀与多个模式匹配，总是选择在Lex程序中先被列出的模式

if

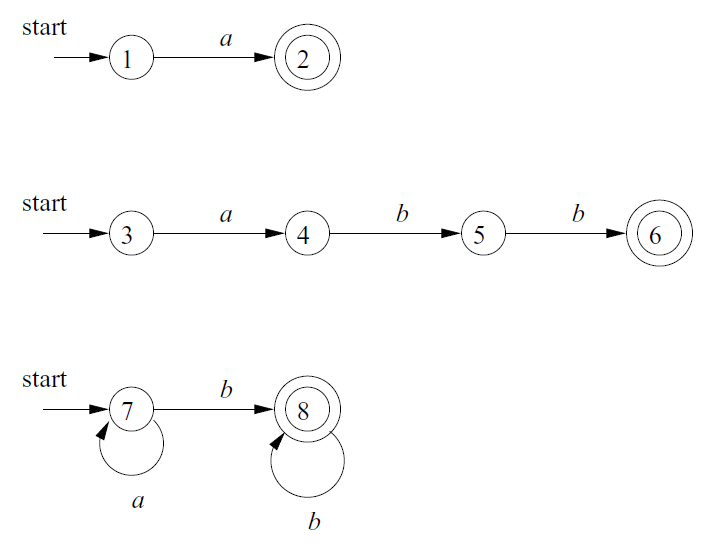


一个Lex程序被转变为有限自动机模拟器使用的转换表和动作

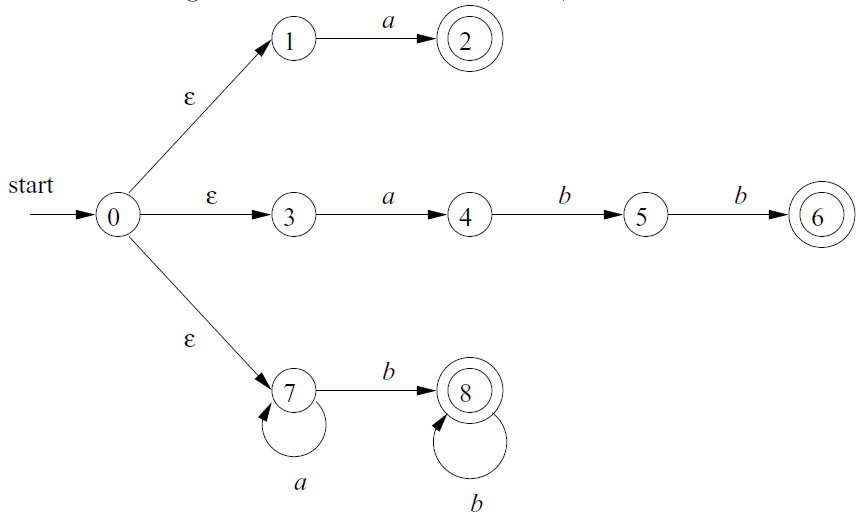
例：

* a {模式P1的动作A1}
* abb {模式P2的动作A2}
* a\*b+ {模式P3的动作A3}

第一步，为每个正则表达式生成NFA



第二步，合并各个正则表达式的NFA



输入串 abba

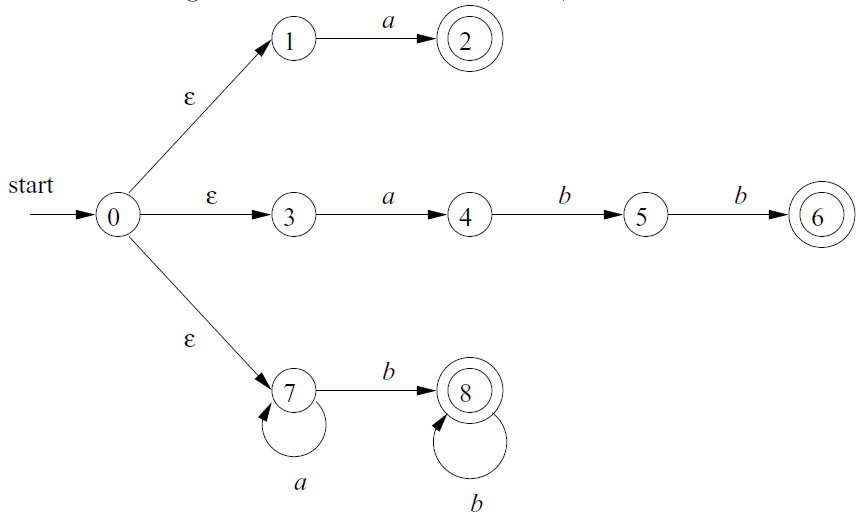
|  |  |
| --- | --- |
| 当前指针 | 当前状态集合 |
| a | {0, 1, 3, 7} |
| b | {2, 4, 7} |
| b | {5, 8} |
| a | {6, 8} |
|  | {} |

输入串abbba

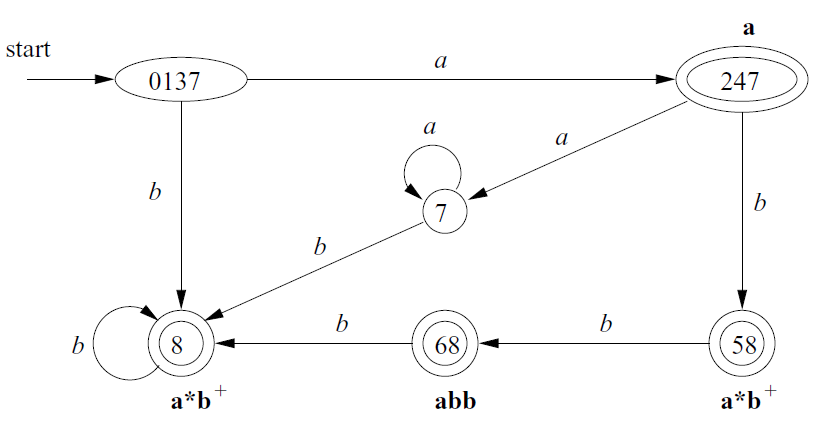
3.8.3 词法分析器使用DFA

例：

* a {模式P1的动作A1}
* abb {模式P2的动作A2}
* a\*b+ {模式P3的动作A3}



由NFA转换为DFA



图中接受状态，都有该状态所标识的模式，该模式即为该状态所关联的模式.

输入串 abba<EOF>

|  |  |
| --- | --- |
| 当前指针 | 当前状态集合 |
| a | {0137} |
| b | {247} |
| b | {58} |
| a | {68} |