从正则表达式直接到DFA

传统方法, 分两步构造DFA (考试内容):

(1) 从正则表达式构造NFA (3.7.4 P100)

McMaughton-Yamada-Thompson算法

一种自底向上方法

(2) 从NFA到DFA的转换 (3.7.1 P96)

子集构造算法

根据正则表达式直接构造DFA (3.9.5 P113)

(1) 对正则表达式r, 扩展其为(r)#，构造出一棵抽像语法树T;

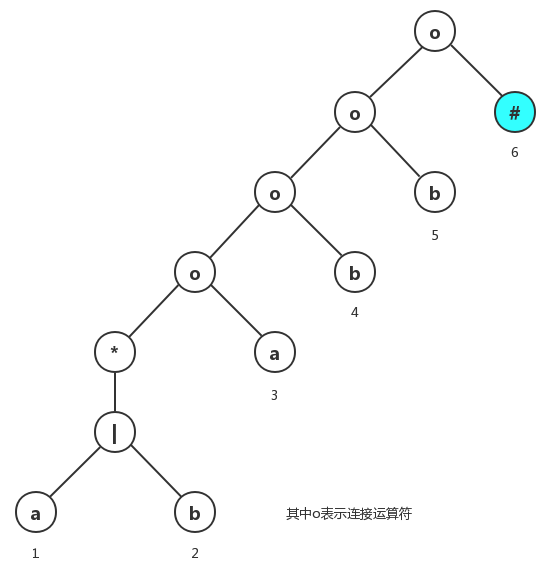
(2) 根据3.9.3和3.9.4节的方法, 计算得到T的函数nullable、firstpos、lastpos和followpos。

(3) 构造出D的状态集Dstats和D的转换函数Dtran

正则表达式r: (a | b)\*abb

扩展后的正则表达式(r)#: (a|b)\*abb#

构造出一棵抽像语法树, 该抽像语法树的叶子结点为字符, 内部结果为操作符: 选择(二元)、拼接(二元)、闭包(一元)



(a|b)\*abb#的抽象语法树

* bool nullable(n)：其中n的取值为抽象语法树中任意结点（可以是叶子结点，也可以是是内部结点。即运算符和运算分量），其返回值的逻辑是当此结点的子表达式中包含有空串ε时返回true，否则为false；
* nodes firstpos(n)：子表达式的语言中某个串的第一个符号，该语言是以n为根的子表达式；
* nodes lastpos(n)：子表达式的语言中某个串的最后一个符号，同样的该语言也是以n为根的子表达式；
* nodes followpos(n)：这相对要复杂一点。比如在抽象语法树中存在两个结点分别为n和m，存在某个串s = a1…aiai+1…ap，即以该正则表达式为模式的串。要满足followpos(n)==m，那么则需要ai和位置n匹配，ai+1和位置m匹配。比如当前有一个符号串aabb，想要构造该串对应于上图抽象语法树中各个节点顺序为1、3、4、5，因此我们就说followpos (3)=4。通俗一点说就是要看抽象语法树指定结点的followpos，需要先找到对应结点在某个符号串中的位置，然后看该符号后一位符号即可。

followposition(p): 在所有的句子中，所有可能出现在p位置符号后的符号的位置集合.

其中表示连接操作。

每个叶子结点为正则表达式中的字母。

每个内部结点为正则表达式中的操作符，并且是一或二元操作符，带一个或两个操作数。

每个结点代表的子树，也是一个正则表达式。

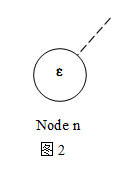
接下来，为每个节点n(即可能是叶子结点，也可能是内部结点)，求nullable(n)、firstpos(n)和lastpos(n).

一、计算nullable(n), 其中n为内部结点/叶子结点

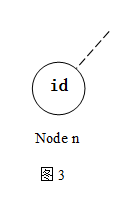
nullable(n)：如果结点n 对应的正则表达式, 能够推导出空串，则nullable(n)为true, 否则为false.

分为5种情况:

　　1、当结点n是叶子结点并且取值为空，此时以n为根结点推导出的句子/串肯定为空，所以此时nullable(n)为true。如图2所示。



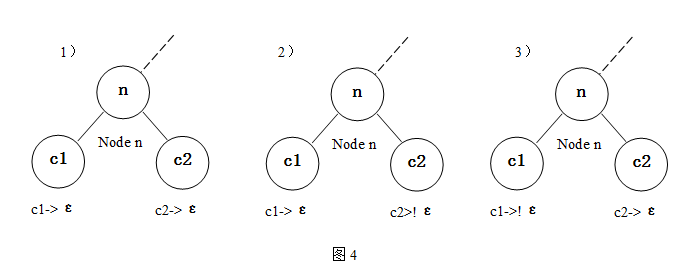
　　2、当结点n是叶子结点并且取值为id，此时以n推导出的句子有非空值id，所以此时nullable(n)为false。



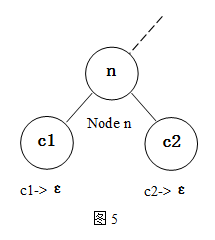
or选择, cat拼接, 二元(两个儿子结点, c1, c2)

star闭包

　　3、当结点n是or结点，此时n结点肯定为内部结点。由于是或运算，当结点n的左子树(c1)或者右子树(c2)能推导出空串时，结点n也能推导出空串。即nullable(n)=nullable(c1) or nullable(c2)。以下三种情况都会使nullable(n)=true。(c1->ε表示c1能推导出空串ε，下同)



　　4、当结点n是cat结点，此时n结点肯定为内部结点。由于是连接运算，当结点n的左子树（c1）和右子树（c2）能同时推导出空串，则结点n能推导出空串，即nullable(n)=nullable(c1) and nullable(c2)。以下一种情况nullable(n)=true。

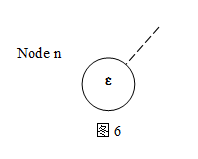


1. 当结点n是star结点，此时n结点肯定为内部结点，由Kleene闭包运算的定义可知结点n推导出的句子集合包括空串，因此nullable(n)=true。

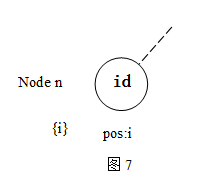
二、计算firstpos(n), 其中n为内部结点/叶子结点

firstpos(n)：以结点n为根的子树中的位置集合。这些位置对应于以结点n为根推导出的某个句子的第一个符号（“某个”代表可能有多个，因此firstpos的计算结果是位置的集合）。我们还是按照nullable的分析方法，以五种结点类型说明firstpos(n)的计算规则。

1、当结点n为叶子结点并且为空串，因为是空串，所以此时没有第一个符号，即firstpos(n)={Ø}。 lastpos(n) = { Ø}

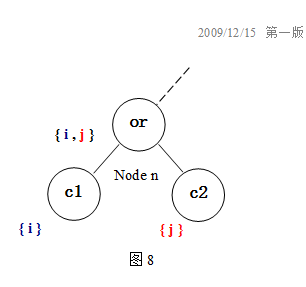


　　2、当结点n为位置i的叶子结点。此时结点n只能推导出位置i的终结符，因此firstpos(n)={i} lastpos(n) = {i}



　　3、当结点n为or结点（内部结点），此时进行or运算，左子树(c1)推导出的第一个位置的集合firstpos(c1)包含于firstpos(n)，同样右子树(c2)推导出的第一个位置的集合firstpos(c2)包含于firstpos(n)。因此firstpos(n)等于左右子树firstpos的并集。即firstpos(n)=firstpos(c1) U firstpos(c2)。

lastpos(n) = lastpos(c1) U lastpos(c2)



　　4、当结点n为cat结点（内部结点），此时进行连接运算，若左子树c1不能推导出空串，那么结点n推导出的某个句子的第一个符号一定在firstpos(c1)中。若左子树c1能推导出空串，那么第一个符号就有可能出现在右子树推导出的句子中。

　　因此：

if(nullable(c1))

firstpos(n)=firstpos(c1) U firstpos (c2)  //图9

else

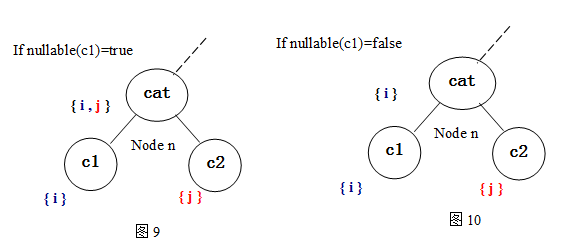
firstpos(n)=firstpos(c1)  //图10

if (nullable(c2))

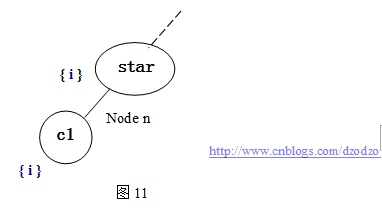
lastpos(n) = lastpos(c1) U lastpos(c2)

else

lastpos(n) = lastpos(c2)



　　5、当结点n为star结点（内部结点），结点n只有一个棵子树(c1)，无论结点n的能不能推导出空串，firstpos(n)=firstpos(c1)。lastpos(n) = lastpos(c1)



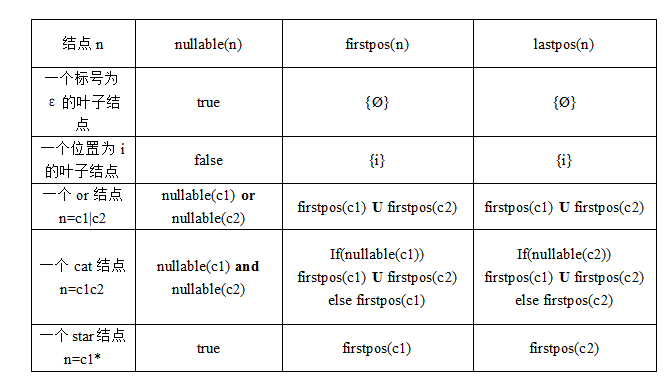
三、计算lastpos(n), 其中n为内部结点/叶子结点

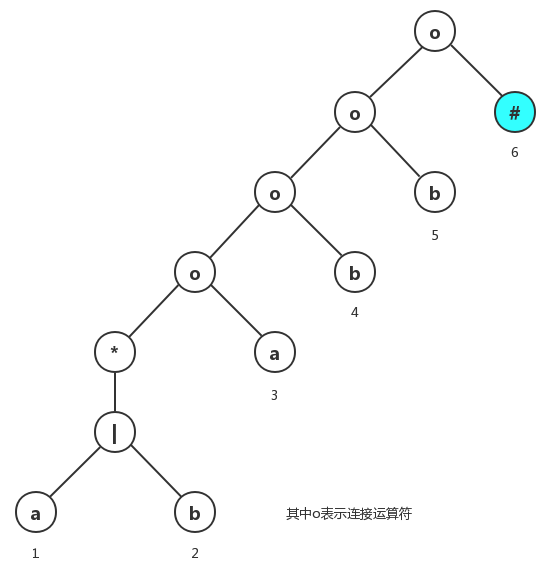
last position(n): 以结点n为根的子树中的位置集合。这些位置对应于以结点n为根推导出的某个句子的最后一个符号（“某个”代表可能有多个，因此lastpos的计算结果是位置的集合）。

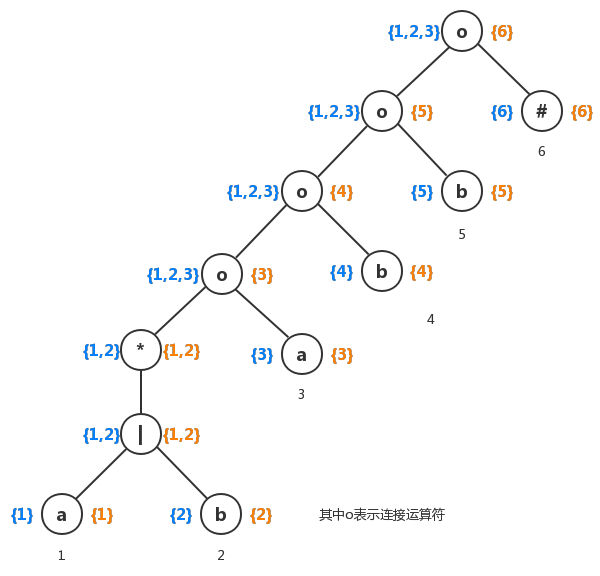
计算lastpos的规则在本质上和计算firstpos的规则相同，但是在针对拼接操作符cat结点的规则中，子树c1和c2的角色要对调。

**总结**

　　nullable(n)、firstpos(n)、lastpos(n)的计算规则总结：







图中蓝色表示firstpos集合，而橙色表示lastpos集合。

|  |  |
| --- | --- |
| 位置i | followpos(i) |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

四、计算followpos(p), 其中p为叶子结点

只有两种情况会使得正则表达式的某个位置跟在另一个位置之后：

followpos(p)定义了一个和位置p相关的、抽象语法树中某些位置相关的集合。

* 当n是一个cat结点，且其左右子树分别为c1、c2，那么对于lastpos(c1)中的所有位置i，firstpos(c2)中的所有位置都followpos(i)中。
* 当n是一个star结点，且i是lastpos(n)中的一个位置，那么firstpos(n)中的所有位置都在followpos(i)中。

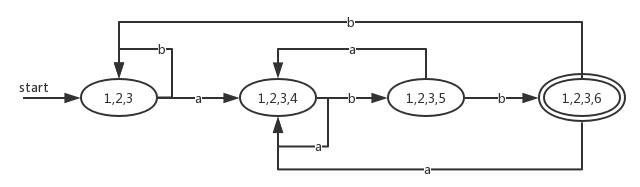
|  |  |
| --- | --- |
| 位置n | followposition(n) |
| 1a | 1, 2, 3 |
| 2b | 1, 2, 3 |
| 3a | 4 |
| 4b | 5 |
| 5b | 6 |
| 6# |  |

DFA初始状态对应的位置集合是firstpos(root) = {1a, 2b, 3a}表示的意思是说, 这个正则表达式生成的句子，第一个符号，一定要么是位置1或2或3。 看作是DFA的初始状态

|  |  |  |  |
| --- | --- | --- | --- |
| 位置集合 | 状态 | a | b |
| {1, 2, 3} | U | {1, 2, 3, 4}B | {1, 2, 3} U |
| {1, 2, 3, 4} | V | {1, 2, 3, 4}B | {1,2,3,5}W |
| {1,2,3,5} | W | {1, 2, 3, 4}B | {1,2,3,6}x |
| {1,2,3,6} | X | {1, 2, 3, 4}B | {1,2,3} U |

DFA: 起始A和终止状态D的确认

那些包含了和结束标记#对应的位置的状态



求(a|b)\*aa的DFA

(1) 画 (a|b)\*aa#的抽象语法树

(2) 为树中每个节点n, 求nullable(n)、firstpos(n)、和lastpos(n)；为树中每个位置p, 求followpos(p)

(3) 构造DFA的状态转换表.