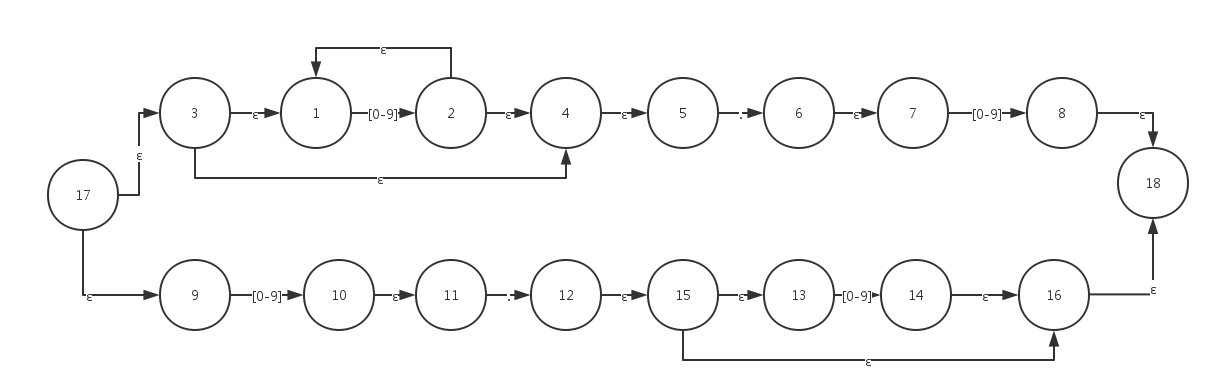
使用NFA状态机识别输入字符串

大家好，欢迎大家来到Coding迪斯尼，前几节，我们实现了正则表达式到NFA有限状态自动机的转换，这几节，我们将致力于NFA转换为DFA, 首先，在这一节，我们看看，如何使用NFA来识别字符串。

给定正则表达式：

D\*\.D | D\.D\*

利用我们前几节开发的程序，对上面的正则表达式构建对应的NFA状态机，构造出的状态机如下（展示原图）：



（上图在文档中可能看不清楚，我将原图传到了附件，大家可以下载来看）

我们将用上面的NFA来识别字符串1.2。

在上图中，最后一个状态18，由于没有出去的边，所以，状态18为接收状态。

我们从状态17开始，记得我们以前说过，对应于ε边，状态机不需要任何输入字符就可以自动进入ε边所指向的下一个状态。由此，从状态17，我们同时可以进入状态：3， 1， 4， 9， 5. 这几个状态组成的集合，确切点说，对应给定的一组初始状态，通过他们的ε边所能达到的状态的集合，我们称之为ε闭包，记做ε-Closure。由于根据初始状态17，通过ε边，我们可以到达的状态有3，1，4，9，因此：

ε-Closure({17}) = { 5,9,17,1,3,4 }

大家注意，ε闭包集合中，包含了给定的初始状态，因为对任意一个状态，它都可以不用输入任何字符就可以抵达它自己，所以，我们默认，每一个点都隐含着一条从自己指向自己的ε边，因此在计算ε闭包时，所得的结果一定包含计算闭包时的输入状态集合，这就是为什么上面的等式右边包含状态17.

从输入中，读入字符1后，得到的跳转状态要从ε闭包集合中找，从图中我们发现，能够读取字符1的状态，是闭包集合中的状态1，9. 从状态1读入字符1后进入状态2，从状态9读取字符1后进入状态10，状态2和10我们称之为转移状态集合，记为：

move({5,9,17,1,3,4}, ‘1’) = {2, 10}

接下来，我们对状态2，10进行ε闭包运算，也就是看看，2，10通过ε边可以抵达哪些状态：

ε-Closure({2, 10}) = { 10,5,2,11,1,4 }.

接下来我们读入字符是 . , 从上面的闭包集合看输入点后的转移集合，根据状态图，我们得知，状态5输入 . 后进入状态6，状态11接收字符 . 后进入状态12， 因此我们有：

move({ 10,5,2,11,1,4 }, '.')= { 6,12 }

然后，我们对转移集合做ε闭包运算有：

ε-Closure( 6,12 ) = { 6,12,7,18,16,13,15 }

大家注意，此时闭包集合中包含了接收状态18，这也意味着，当前输入的字符所组成的字符串 1. 是可以被我们的状态机接受的。但是由于输入还没有处理完，因此我们继续将没处理的字符输入状态机中，当前要输入的字符是2，于是：

move({ 6,12,7,18,16,13,15 }, '2')= { 14,8 }

对转移集合{14,8}做闭包运算有：

ε-Closure( 14,8 ) = { 14,8,18,16,13 }

此时，闭包集合中包含接受状态18.

此时，所有字符都已经处理，同时状态机进入了接受状态，因此字符串1.2可以被我们的NFA状态机所接受。

这一节，我们简单描述了NFA有限状态自动机对输入字符串的识别流程，在下一节，我们将用代码实现这一节所描述的识别算法。

本节附件中所提供的代码已经实现了本节所讲的识别算法，大家可以先把代码下下来，跑起来，得到一些感性认识，下一节，我将以调试运行的方式，向大家展现代码的实现细节。