有穷自动机（Finite Automate）是用来模拟实物系统的数学模型，它包括如下五个部分：

* 有穷状态集States
* 输入字符集Input symbols
* 转移函数Transitions
* 起始状态Start state
* 接受状态Accepting state(s)

有穷自动机，按照转移函数的不同，又可分为确定型有穷自动机（Determinism Finite Automate, DFA），与非确定型有穷自动机（Non-determinism Finite Automate, NFA）。  
非确定有穷自动机容许转移函数不确定，换句话说，对任意状态，输入任意一个字符，可以转移到0个，1个或者多个状态

正则表达式就是建立在自动机的理论基础上的：用户写完正则表达式之后，正则引擎会按照这个表达式构建相应的自动机（可能是NFA，也可能是DFA，但它们必定是等价的），若输入一串文本之后，自动机抵达了接受状态，则这串文本可以“匹配”用户指定的正则表达式。

DFA比较快，但不提供Backtrack（回溯）功能，NFA比较慢，但提供了Backtrack功能。

DFA引擎在任意时刻必定处于某个确定的状态，而NFA引擎可能处于一组状态之中的任何一个，所以，NFA引擎必须记录所有的可能路径（trace multiple possible routes through the NFA），NFA之所以能够提供Backtrack的功能，原因就在这里。

正则引擎主要可以分为两大类：一种是DFA，一种是NFA。这两种引擎都有了很久的历史(至今二十多年)，当中也由这两种引擎产生了很多变体！于是POSIX的出台产生规范了不必要变体的继续产生。这样一来，目前的主流正则引擎又分为3类：一、DFA，二、传统型NFA，三、POSIX NFA。

POSIX NFA 引擎与传统的 NFA 引擎类似，不同的一点在于：在它们可以确保已找到了可能的最长的匹配之前，它们将继续回溯。因此，POSIX NFA 引擎的速度慢于传统的 NFA 引擎；并且在使用 POSIX NFA 时，您恐怕不会愿意在更改回溯搜索的顺序的情况下来支持较短的匹配搜索，而非较长的匹配搜索。

## Dfa nfa引擎的使用情况表

　　目前使用DFA引擎的程序主要有：****awk,egrep,flex,lex,MySQL,Procmail****等；  
　　使用传统型NFA引擎的程序主要有：****GNU Emacs,Java,ergp,less,more,.NET语言,PCRE library,Perl,PHP,Python,Ruby,sed,vi****；  
　　使用POSIX NFA引擎的程序主要有：****mawk,Mortice Kern Systems’ utilities,GNU Emacs****(使用时可以明确指定)；  
　　也有使用DFA/NFA混合的引擎：****GNU awk,GNU grep/egrep,Tcl****。

## 举例简单说明NFA与DFA工作的区别：

　　比如有字符串****this is yansen’s blog****，正则表达式为 ****/ya(msen|nsen|nsem)/**** (不要在乎表达式怎么样，这里只是为了说明引擎间的工作区别)。

　　NFA工作方式如下，先在字符串中查找 ****y**** 然后匹配其后是否为 ****a**** ，如果是 ****a**** 则继续，查找其后是否为 ****m**** 如果不是则匹配其后是否为 ****n**** (此时淘汰****msen****选择支)。然后继续看其后是否依次为 ****s,e****，接着测试是否为 ****n**** ，是 ****n**** 则匹配成功，不是则测试是否为 ****m**** 。为什么是 m ？因为 NFA 工作方式是以正则表达式为标准，反复测试字符串，这样同样一个字符串有可能被反复测试了很多次！

而DFA则不是如此，DFA会从 ****this****中 ****t**** 开始依次查找 ****y****，定位到****y**** ，已知其后为 ****a**** ，则查看表达式是否有 ****a****，此处正好有 ****a**** 。然后字符串 ****a****后为 ****n**** ，DFA依次测试表达式，此时 ****msen****不符合要求淘汰。****nsen**** 和 ****nsem**** 符合要求，然后DFA依次检查字符串，检测到****sen**** 中的****n**** 时只有****nsen****分支符合，则匹配成功！

## 但是NFA以表达式为主导，反而更容易操纵，因此一般程序员更偏爱NFA引擎！

　　由此可以看出来，两种引擎的工作方式完全不同，一个(NFA)以表达式为主导，一个(DFA)以文本为主导！一般而论，DFA引擎则搜索更快一些！但是NFA以表达式为主导，反而更容易操纵，因此一般程序员更偏爱NFA引擎！

## 两种引擎各有所长，而真正的引用则取决与你的需要以及所使用的语言！

### 为什么要将NFA转化为DFA？

如题，

[编译](http://iask.sina.com.cn/c/1069.html" \t "http://iask.sina.com.cn/b/_blank)

原理中为什么要将NFA转化为DFA？

[s89gfaf](http://iask.sina.com.cn/user/see/53b534b711e0ee3bf7abbf84-IA-1.html" \t "http://iask.sina.com.cn/b/_blank) | 12-05-10 标签： [理中](http://iask.sina.com.cn/word/56e909fd0cf230ced409f1e8.html)

### **好评回答**

[IMG_256](http://iask.sina.com.cn/user/pcsee/53b526ca11e0ee3bf786a5c1-IA-1.html)

[bv1200](http://iask.sina.com.cn/user/pcsee/53b526ca11e0ee3bf786a5c1-IA-1.html" \t "http://iask.sina.com.cn/b/_blank) | 12-05-10 **0** **0** [举报](http://iask.sina.com.cn/b/javascript:void(0);)

编译原理中DFA是确定的有限自动机，而NFA是非确定有限自动机，将NFA化为DFA是将状态数减少，更为简单确定

希望能给你帮助。

### **其他答案**

（共 1个回答）

因为NFA是一种状态不确定的自动机，所以这种自动机不便机器实现；  
DFA是有限确定状态的自动机，它的状态转换的条件都很确定，所以它比较方便机器实现，同时在识别能力也和NFA相当（

[书](http://iask.sina.com.cn/c/213.html" \t "http://iask.sina.com.cn/b/_blank)

上已经证明了每一种NFA都可转换为同样识别能力的DFA），所以转换为DFA是更利于实现的

正则表达式引擎及其分类\_知识库\_博客园.html