Atitit 正则表达式原理与概论

在1956 年,一位名叫Stephen Kleene的数学科学家，他在Warren McCulloch和Walter Pitts早期工作的基础之上，发表了一篇题目是《神经网事件的表示法》的论文，利用称之为正则集合的数学符号来描述此模型，引入了正则表达式的概念。正则表达式被作为用来描述其称之为“正则集的代数”的一种表达式，因而采用了“正则表达式”这个术语。

之后一段时间，人们发现可以将这一工作成果应用于其他方面。Ken Thompson就把这一成

自此以后，正则表达式被广泛地应用到各种UNIX或类似于UNIX的工具中，如大家熟知的[Perl](http://baike.baidu.com/view/46614.htm" \t "http://baike.baidu.com/_blank)。Perl的正则表达式源自于Henry Spencer编写的regex，之后已演化成了pcre（Perl兼容正则表达式Perl Compatible Regular Expressions），pcre是一个由Philip Hazel开发的、为很多现代工具所使用的库。正则表达式的第一个实用应用程序即为Unix中的 qed 编辑器。

## 引擎

[编辑](http://baike.baidu.com/javascript:;)

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的出台规避了不必要变体的继续产生。这样一来，主流的正则引擎又分为3类：一、DFA，二、传统型NFA，三、POSIX NFA。

[DFA](http://baike.baidu.com/view/289418.htm" \t "http://baike.baidu.com/_blank) 引擎在线性时状态下执行，因为它们不要求回溯（并因此它们永远不测试相同的字符两次）。DFA 引擎还可以确保匹配最长的可能的字符串。但是，因为 DFA 引擎只包含有限的状态，所以它不能匹配具有反向引用的模式；并且因为它不构造显示扩展，所以它不可以捕获子表达式。

传统的 NFA 引擎运行所谓的“贪婪的”匹配[回溯算法](http://baike.baidu.com/view/6056523.htm" \t "http://baike.baidu.com/_blank)，以指定顺序测试正则表达式的所有可能的扩展并接受第一个匹配项。因为传统的 NFA 构造正则表达式的特定扩展以获得成功的匹配，所以它可以捕获子表达式匹配和匹配的反向引用。但是，因为传统的 NFA 回溯，所以它可以访问完全相同的状态多次（如果通过不同的路径到达该状态）。因此，在最坏情况下，它的执行速度可能非常慢。因为传统的 NFA 接受它找到的第一个匹配，所以它还可能会导致其他（可能更长）匹配未被发现。

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。

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！

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！

（摘自《[正则表达式之道](http://baike.baidu.com/view/1427403.htm" \t "http://baike.baidu.com/_blank)》）