Java 正则表达式优化

Regular Expression Optimization in Java

葛汉强　陈和平　(武汉科技大学 信息科学与工程学院 湖北 武汉 430081)

摘　要：正则表达式是处理文本的有利工具,如何构造一条优化的正则表达式是一个关键的问题。作者在详细分析相关原理的基础上提出解决该问题的"较优化原则",并通过一个具体实例的构造过程详细说明了该原则,最后给出了一个 Java 测试程序。"较优化原则"在使用海量数据的 Web 程序中有着较广泛的应用前景,使程序的运行效率有了较大地提高。

关键词：正则表达式 NFA 引擎 回溯 匹配模式 Java

正则表达式是用某种模式去匹配一类字符串的一个公式,它能够以简单通用的方式来实现匹配、搜索、替换等强大的功能,在处理文本的程序中有着广泛的应用。PHP、C#和 Java 等编程语言都支持正则表达式。Java 在 jdk1.4 中引入了 java.util.regex 包以实现对正则表达式的支持。

在编写正则表达式的程序中,构造一个优化的正则表达式无疑是所希望的。可是,在处理实际问题时,符合相关条件的正则表达式往往会有多种表达形式,编程人员所面临的问题就是如何去构造一个"优化的"正则表达式。作者在分析相关原理的基础上给出了解决该问题的"较优化原则",并通过构造匹配电子邮箱地址正则表达式的实例对该原则进行了说明,最后给出了一个测试程序。

1　相关原理

1.1　NFA 定义

非确定有限自动机(Nondeterministic Finite Automaton)简称 NFA[3],由一个五元式 M = (S, ∑, δ, S_0, F) 组成,其中:

(1) S 是一个有限集,它的每个元素称为一个状态。

(2) ∑是一个有穷字母表,它的每个元素称为一个输入字符。

(3) δ是一个从 S × ∑ * 到 S 的子集的映射(亦称为状态转移函数),即 $\delta: S \times \sum^* \rightarrow 2^S$。

(4) S0 是一个非空初态集。

(5) F 是一个终态集(可空),F ∈ S。

由定义可知,NFA 的状态转换函数是非单值映射的,即从当前状态输入一个字符后可以转换到多个状态。自动机初始状态为 S0,逐一读入输入字符串中的每一个字母,根据当前状态和读入的字母,由状态转移函数 δ 控制进入下一个状态。如果输入字符串读入结束时自动机的状态属于结束状态集合 F,则说明该自动机接受该字符串,否则为不接受。

1.2　NFA 引擎(Engines)

Java 的正则表达式是基于 NFA 引擎[1]的匹配。所谓引擎是指基于 NFA 原理构建的 Java 的正则包。在 NFA 引擎的匹配过程中,目标文本中的某个字符可能会被正则表达式中的不同部分重复检测,即使某个字符表达式可以匹配,为了检查表达式剩余部分,也可能需要再一次(或多次)检测,所以直到抵达正则表达式的末尾,也无法确定全局匹配是否成功。换句话说,NFA 是以表达式为主导的匹配方式,不同的表达式的构造方式对引擎的效率有着重要的影响。

1.3　回溯(Backtracing)

当进行正则表达式匹配时,有着多种选择途径,在按其中一条途径进行匹配时,会同时记录着另一条备份途径,如果发生了不匹配,则会回溯[1]到另一条备份途径的位置,选择其它备用的途径进行尝试,直到尝试完所有的备份途径为止,因此,NFA 引擎的效率取决于回溯的次数。

1.4 两条规则

规则一：优先选择最左端匹配

字符的匹配是从左向右进行的。即使后面有可能发现一个“更好”的匹配，NFA 引擎也总是返回最左边的匹配。

规则二：标准匹配量词的贪婪性（Greedy）

标准匹配量词如：?、*、+、{n,m} 等字符。贪婪性是指尽可能地匹配较多的字符。

1.5 正则表达式的构造语法

为了说明相关问题需要，在表 1 中列出了常用的部分正则表达式的构造语法，其它详细说明见 jdk API 1.6.0 的 java.util.regex 包文档[2]。

表 1　正则表达式的构造语法

名　称	构　造	匹配含义
欲定义字符类	.	任何字符
欲定义字符类	\d	数字[0-9]
欲定义字符类	\s	空白字符
欲定义字符类	\w	单词字符[a-z A-Z 0-9]
字符类	[a-zA-Z]或[a-z[A-Z]]	a~z 或 A~Z 的一个字符
字符类	[^X]	除了 X 的任何字符
边界匹配器	^	行的开头
边界匹配器	$	行的结尾
边界匹配器	\b	单词边界
Greedy 数量词	X?	X，一次或一次也没有
Greedy 数量词	X*	X，零次或多次
Greedy 数量词	X+	X，一次或多次
Greedy 数量词	X{n,m}	X，至少 n 次，不超过 m 次
Logica 运算符	XIY	X 或者 Y
非捕获组	{? =X}	是 X，向前看零宽度的非捕获

2 较优化原则

由上述分析可知，构造一个较优化的正则表达式通常需要考虑到两方面的因素：即表达式如何有效满足实际问题的条件和如何有效提高 NFA 引擎的匹配效率。前一问题是指正则表达式的匹配模式能将符合要求的字符串完全匹配出来，而后一问题的关键是能否减少生成匹配模式的次数。

通过元组字符 .、?、*、+、{n,m}、[a-zA-Z] 等可以发现，匹配模式的次数是由于元组字符的不确定性产生的。如".*"表示任意的字符串，虽然有很强的通用性，但可能生成大量的匹配模式。为了减少生成匹配模式的次数，需要将实际问题中的条件进一步地细化，用比较确定的字符组合，但这是以牺牲匹配模式的通用性为代价的。所以，在处理实际问题的过程中，需仔细分析要达到的目标要求，在通用性和减少次数两方面加以权衡，在满足目标要求的前提下，尽量减少匹配模式的生成次数。

严格说来，提高引擎的效率最有效的办法是对引擎的优化，这在实际的处理过程中很难做到，同时也没有必要。在 jdk API 1.6.0 的 java.util.regex 包中已经封装好了相应的类，编程时可直接引用。因此，本文提到的提高引擎效率是指遵循引擎原理下的对表达式的优化构造，目标是减少回溯次数。达到该目标的一个较好的方案是采用非捕获组，作者在实例中采用的就是向前查看零宽度的非捕获组。因为是零宽度非捕获，在匹配过程中并不消耗字符，也不需要记录备用的途径，同时由于本身是一个判断条件，在匹配过程中使得引擎失败几率加快，从而可有效地阻止大量回溯。

生成匹配模式的次数减少，相应的回溯次数也就减少，所以两者是内在相关的。

综上所述，较优化原则就是在满足实际要求和遵循引擎原理的前提下，以减少匹配生成模式次数和回溯次数为目标来构造正则表达式的一种优化策略。

3 实例

通过一个实例进一步说明较优化原则和优化过程。编写能自动提取某一网页中所有的 E-mail 地址的程序时，我们需要构造一个电子邮箱地址格式的正则表达式。

TCP/IP 体系的电子邮件系统规定电子邮箱地址的格式如下：收信人邮箱名@ 邮箱所在主机的域名

若用".+"作为“收信人邮箱名”匹配模式，表面上可满足要求，实际效果却不佳。由于".+"的不确定性不仅增加了匹配次数，同时还会将一些不是收信人邮箱名的其它字符也包括进来，达不到实际问题的要求。

对实际问题中的条件作进一步的细化可以发现：“收信人邮箱名”通常是由英文字母、数字、下划线、点和横线构成，若将匹配模式改为"[\w[.-]]+"，范

围更加明确,匹配模式的匹配次数相应地减少。

再则,通常"收信人邮箱名"是以字母打头,将\w、[.-]的排列次序交换,依据规则一的描述[.-]靠左先匹配,这样失败匹配几率加大,加速引擎失败,故将匹配模式改为"[[.-]\w]+"后,可进一步提高匹配效率。

同时,由于注册"收信人邮箱名"字符并不是无限的,一般限定在3-35个字符内,所以可以进一步将匹配模式改写为"[[.-]\w]{3,35}"。

进一步地,可以得到一个通用的电子邮箱的地址匹配模式:"[[.-]\w]{3,35}@[[.-]\w]+\.[\w]+"(对于"邮箱所在主机的域名"可作同理分析,不再详叙)。

注意到满足电子邮箱地址的格式中必须包含@字符,使

用向前查看零宽度非捕获组后的表达式是:"[[.-]\w]{3,35}{?=@}@[[.-]\w]+\.[\w]+"(1)

如果程序使用局限是国内的E-mail地址,可以根据国内的"邮箱所在主机的域名"的构造规则将表达式改写为:

"[[.-]\w]{3,35}{?=@}@[[.-]\w]+\.com(\.cn)?"(2)

通过该实例可知,正则表达式的构造并没有一个通用的模式,其构成很大程度上依赖于需处理问题的条件要求,无法得到一个满足所有条件的式子,所得到的较优化的表达式只是一个满足相关条件的优化式子。例如,国内网页论坛上某人留的E-mail地址是xxx@163.copm,我们立刻可以判断它是一个无效的地址,可是(1)式仍会认为是一个有效的地址而匹配成功,虽然(1)式是一个比(2)式更通用的式子,但在不同的要求下,也许并不是个更优化的式子。

4 测试程序

以下给出一个上述实例的Java测试程序。

```
import java.io.*;
import java.net.URL;
import java.net.URLConnection;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class Emailaddress {
    public static void main(String[] args) throws IOException {
        String urlLink = "http://gceclub.sun.com.cn/...." ;
                                                    //连接地址
        URL url = new URL(urlLink);
        URLConnection conn = url.openConnection();
        conn.connect(); //创建连接
        InputStream is = conn.getInputStream();
        BufferedReader br = new BufferedReader(new InputStreamReader(is,"gbk"));
        String line;
        while((line = br.readLine())!=null){ //读网页源代码
        parse(line);
        }
        is.close();
        br.close();
    }
    private static void parse(String line){ //提取E-mail地址
        Pattern p = Pattern.compile("[[.-]\w]{3,35}(?=@)@[\w[.-]]+\.com(\.cn)?");
        Matcher m = p.matcher(line);
        while(m.find()){
          System.out.println(m.group());
        }
    }
}
```

该程序是用来检测构造的正则表达式是否正确,而非用于评测构造的正则表达式的优化效率。实际上,只有用海量的数据才可能观察出两个正则表达式的优化的效率差异,对于少量的数据,区别不是太大。

5 结束语

本文提出的"较优化原则"是遵循Java正则表达式相关原理上的逻辑推导而得到的,(下转第32页)

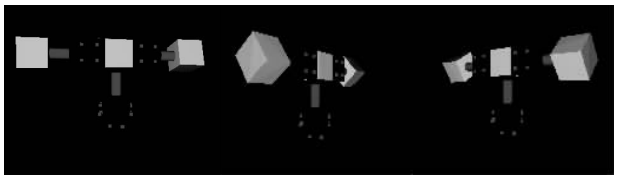

图 10　模块组合仿真实验

本体的信息交换和运动同步。仿真系统的界面如图 8 所示。图 9 显示单个模块运动的过程。图 10 显示由 3 个模块机器人组合在一起的机器人系统的运动过程。

5　结论

本文就可重构模块化机器人以及通用接口技术作了初步的研究，设计了一种新型的单转动自由度阵列式的三维机器人模块，该模块兼具阵列式和串联式机器人系统的优点，通用性高，结构简单，控制容易。设计了一种新型平面连接机构和电气连接部件；分析了机器人控制系统功能和要求，在此基础上提出控制系统总体构建；建立了机器人的实验系统，仿真机器人构形和运动，验证了模块设计的正确性和整体运动构形规划方法的有效性。

对于今后的研究工作有以下几点展望：

1. 针对具体的应用背景进行模块结构的改进设计，提高完成任务的多样行和实用性。

2. 结合人工智能领域的研究成果和多传感器信息融合技术，使其具有更高的智能。

3. 利用仿生学的研究成果，结合模块化机器人的结构特点，研制可重构模块化仿人机器人。

4. 采用蓝牙无线通讯，以适应对通讯速度和质量要求更高的应用场合。

参考文献

1　龙斌，毛立民，等. 国外自主变结构模块机器人发展现状. 机械设计，2005，22(5)：1－3.

2　Jinxiang Shen, Jiangping Liu, et al. Distributed Control System for Modular Self－Reconfigurable Robots. Proc. of the 2003 IEEE, International Conference on Robotics, Intelligent Systems and Signal Processing, Changsha, China － October 2003：932－9.

3　刘明尧，谈大龙，李斌. 可重构模块化机器人现状和发展. 机器人. 2001，23(3)：275－279.

4　K. Stoy, W. M. Shen, P. M. Will. A simple approach to the control of locomotion in self－reconfigurable robots. Robotics and Autonomous Systems, 2003：191－199.

5　冯金光，周华平，马宏绪. 基于 CAN 总线和 DSP 的仿人机器人运动控制系统研究. 总线与网络，2004，5：29－33.

6　D. N. Ly, et al. A modular and distributed embedded control architecture for humanoid robots. Proc. of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 28 － October 2, 2004, Sendai, Japan：2775－2780.

7　T. Taira, et al. Design and implementation of reconfigurable modular humanoid robot architecture. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. (IROS 2005) 3566－3571.

(上接第 104 页)

并给出了一个如何构造正则表达式的具体实例。基于 web 的应用常常需要构造正则表达式去匹配海量的数据，“较优化原则”提供了一种针对性较强的可行方法，可提高程序运行的效率。

虽然本文仅介绍了基于 Java 语言的正则表达式的构造原则，其思路对于在其他语言下构造优化的正则表达式有一定的参考作用。

参考文献

1　Jeffrey E. F. Friedl. Mastering Regular Expressions. Publisher：O'Reilly. 2006：146－166.

2　http://gceclub.sun.com.cn/Java_Docs/jdk6/html/zh_CN/api/index.html 技术文档 2007.

3　陈火旺编译原理. 北京：国防工业出版社，2000.