AutomatonConversion项目文档

计研091 黄崇迪 2009210940

huangcd.thu@gmail.com

January 21, 2010

目录

1 项目说明

* 组号：21（对应GoogleCode中trunk目录下的21Conversion文件夹）

* 成员：计研091 黄崇迪 2009210940 hcd05@mails.tsinghua.edu.cn

1.1 项目开发环境

操作系统	Windows Vista^{TM} Home Premuim Server Pack 2
CPU	Intel(R) Core(TM)2 Duo CUP P7350 @ 2.00GHz × 2
内存	2.00 GB
JDK版本	1.6.0_13
开发工具	IntelliJ IDEA 8.1.4

1.2 项目依赖类库

项目依赖于JUNG 2.0和Dom4j 1.6两个类库。最多包含下列jar文件：
dom4j.jar collections-generic-4.01.jar jung-visualization-2.0.jar
jung-algorithms-2.0.jar jung-graph-impl-2.0.jar jung-api-2.0.jar
concurrent-1.3.4.jar stax-api-1.0.1.jar colt-1.2.0.jar wstx-asl-3.2.6.jar

2 程序使用说明（Quick Start）

2.1 基本使用

通过bin目录下面的run.bat文件或者AutomatonConversion.jar启动程序。启动后通过**open xml automaton**按钮打开XML文件。图1是程序运行某个时刻的截图及各个按钮的简要使用说明。

AutomatonViewer部分显示自动机，初始状态用深蓝色表示，终止状态用粗框表示，其他状态用浅蓝色表示。转换以边的形式表示（如果一个

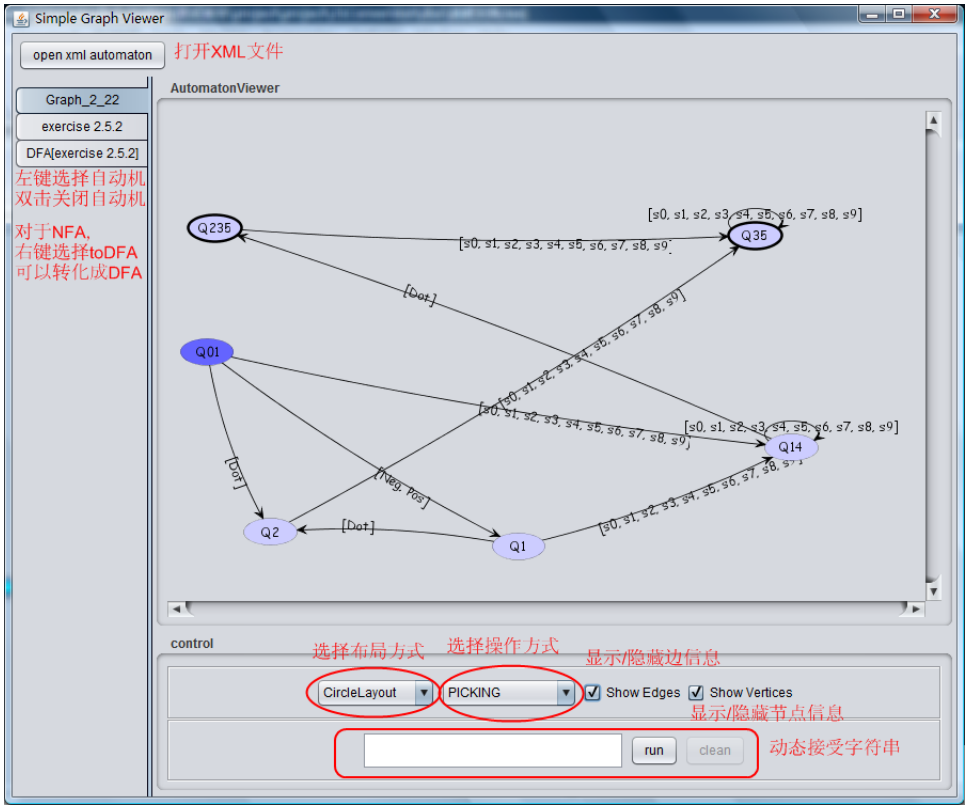

Figure 1: 程序主题界面及简要使用说明

状态可以通过多个符号转换成另一个状态，那么图中把多个转移合成为一条边）。

AutomatonViewer部分支持鼠标中键缩放，图形拖动（TRANSFORMING状态）和节点拖动（PICKING状态）等功能。 鼠标在节点或者边上悬停出现详细信息。 也可以选择不同的布局方式（但目前没有效果比较好的布局方式）。

目前程序的功能主要针对DFA实现。因此对于NFA，动态接受字符串部分不可用， 但可以通过右键点击标签选择toDFA选项来把NFA转换成DFA。

2.2 动态接受字符串

目前动态接受字符串只针对DFA实现。使用的方法是在输入框中输入字符串 （字符之间需要用“|”分割），然后点run执行。

接受过程中，当前状态在AutomatonViewer中显示为黄色，当前符号在输入框中显示为蓝色。 如果字符串被拒绝，状态和符号都被显示为红色。如果字符串被接受，接受的状态被显示为绿色。

3 算法实现及代码介绍

自动机的表示

自动机在程序里面被表示成状态的集合。转移被表示为状态的属性。 程序里面使用FiniteAutomaton表示抽象的自动机， DFA和NFA继承FiniteAutomaton分别表示相应的自动机。 状态同样对应地有 DFAState和NFAState（实现State接口）。

NFA到DFA的转换

根据课本上的算法实现，从初始状态开始根据传递闭包的转移逐步构造整个DFA。 在NFA.toDFA()函数中实现。

判断自动机是否为空

通过广度优先搜索检查从开始状态是否可达终止状态。 在 DFA.isEmpty() 函数中实现。对于NFA，通过NFA.toDFA().isEmpty()实现。

判断自动机是否为无穷

通过深度优先搜索检查从开始状态到终止状态的路径中是否存在环。在DFA.isInfinite()函数中实现。对于NFA，通过 NFA.toDFA().isInfinite() 实现。

判断字符串时候被接受

根据输入符号遍历状态，字符串结束的时候恰好到达终结状态， 说明字符串被接受。在DFA.accept()函数中通过调用 DFAState.shift() 实现。对于NFA，同样通过 NFA.toDFA().accept() 实现。

XML和自动机的相互转换

根据FA.dtd的描述，调用dom4j进行解析或写出。具体实现在 automaton.io.xml包中的几个类。

显示

利用jung实现，具体实现见graph包和ui包。

简单使用例子

见sample.Test。

4 代码量统计（放大查看）

Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
FAViewer.java	543	453	83%	45	8%	45	8%
DefaultXMLAutomatonReader.java	378	334	88%	19	5%	25	7%
Main.java	195	157	81%	18	9%	20	10%
NFAState.java	186	138	74%	18	10%	30	16%
NFA.java	168	131	78%	12	7%	25	15%
DFAState.java	156	114	73%	17	11%	25	16%
DefaultXMLAutomatonWriter.java	142	111	78%	14	10%	17	12%
Util.java	137	97	71%	28	20%	12	9%
FiniteAutomaton.java	134	101	75%	11	8%	22	16%
DFA.java	126	101	80%	10	8%	15	12%
Test.java	80	59	74%	8	10%	13	16%
TransitionEdge.java	68	50	74%	6	9%	12	18%
State.java	50	12	24%	30	60%	8	16%
XMLAutomatonReader.java	40	21	52%	5	12%	14	35%
UnconvertableException.java	23	14	61%	5	22%	4	17%
XMLAutomatonWriter.java	17	8	47%	5	29%	4	24%
AutomatonGraph.java	13	6	46%	5	38%	2	15%
StateType.java	10	4	40%	5	50%	1	10%
Total:	2466	1911	77%	261	11%	294	12%

5 项目目标及实现

所选项目为**NFA/DFA的转换工具**。具体要求如下：

1. 用Java编写，保证可扩展性。

2. 检查某个字符串是否被指定的自动机接受。

3. 判断语言是否为空，是否为无穷。

4. 实现NFA到DFA的相互转换。

除了实现上述要求内容以外，额外实现了一些功能，具体包括：

1. XML到自动机的相互转换

2. 自动机的可视化及动态接受字符串的过程

6 目前已知的bug

当前版本的程序里面，当打开一个NFA，然后右键转换为DFA的时候。界面上某个状态可能出现多于1次。初步估计这是Jung库存在的bug（或者是不熟悉Jung的使用导致）， 而与NFA→DFA转换算法无关。由于这与项目要求没有太多关联，因此这里不做修正， 而仅仅提供一个折中的方案：每次执行NFA→DFA的同时，程序将DFA输出到程序所在目录； 因此可以重新启动一个程序实例打开输出的DFA。

另外，在同一个程序实例里面打开多个自动机实例也可能导致上面问题。

7 运行结果展示及说明

以课本图2.18（xml实例为data目录下的NFA2.xml）为例进行展示：

程序启动以后打开NFA2.xml文件，程序截图如图2，默认的布局方式是CircleLayout， 相对来说比较清晰。拖动节点（PICKING状态）重新布局得到图3。

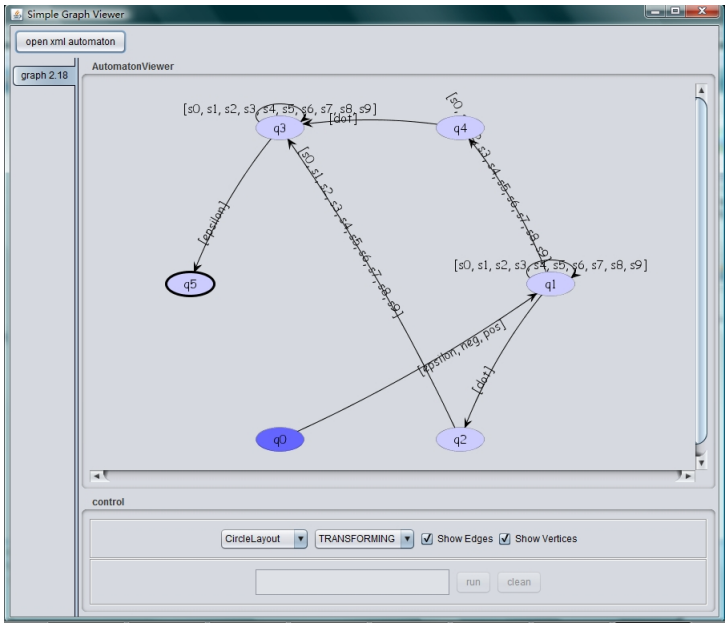

Figure 2: NFA

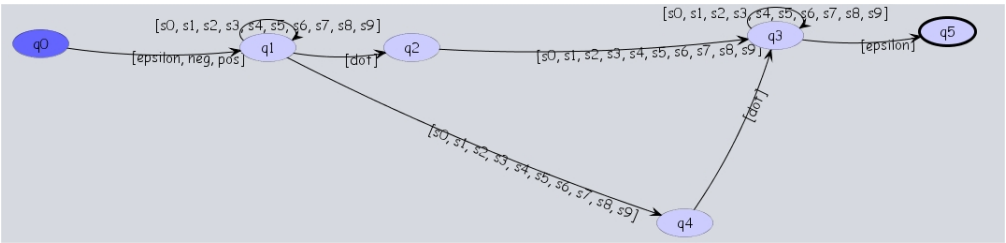

Figure 3: NFA2

右键点击标签选择toDFA，然后重新启动程序，打开目录下新生成的DFA[graph 2.18].xml文件， 拖动节点重新布局得到图4，与课本图2-22比较可以发现， 图4仅仅多了一个空状态收集不接受的转移，这说明转移算法是正确的。

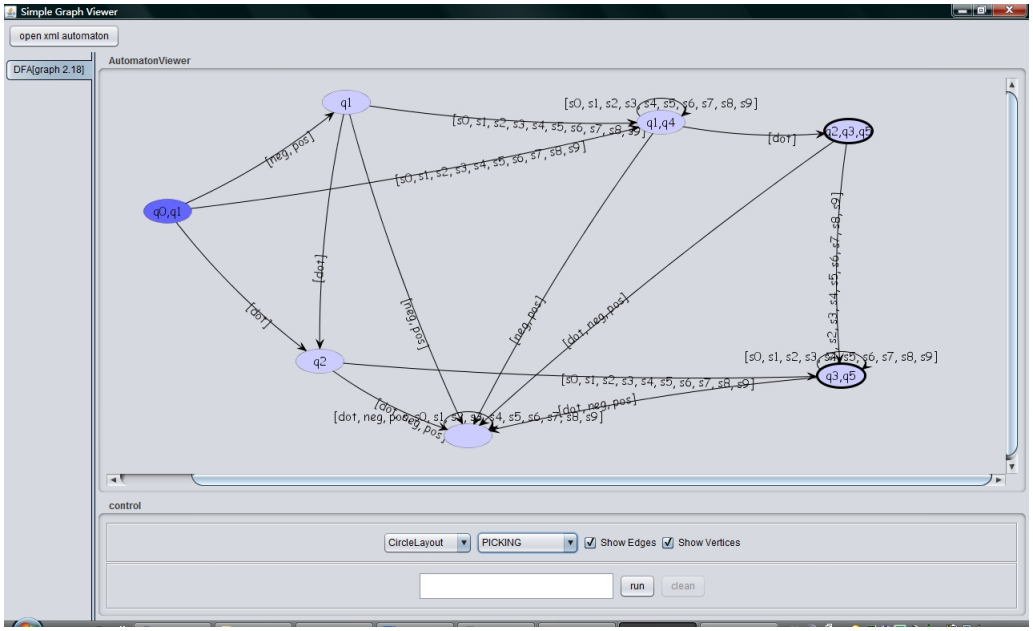

Figure 4: DFA

输入一个正确的例子和错误的例子进行检验：

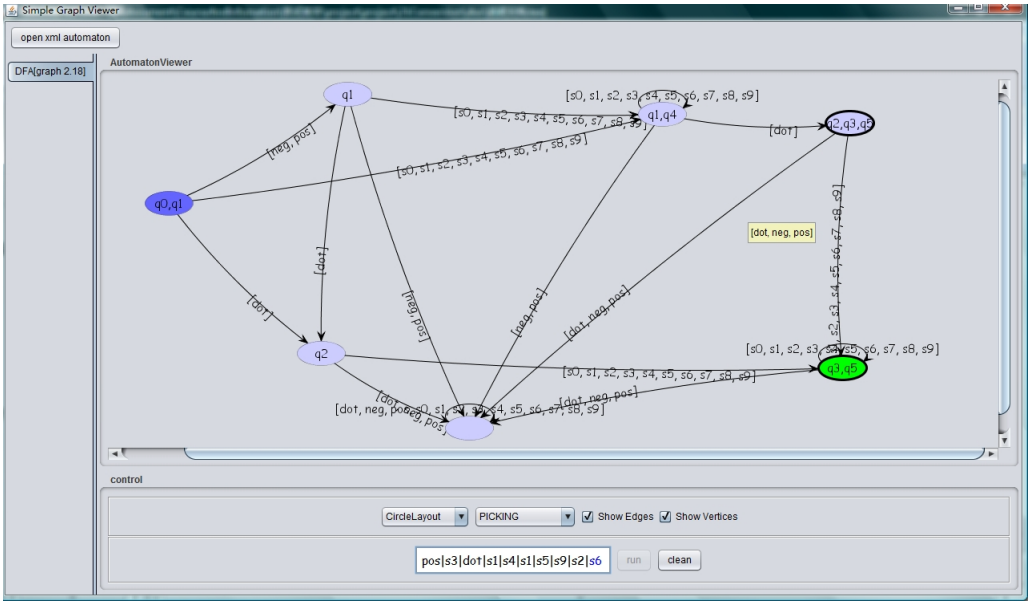

Figure 5: 正确的例子（+3.1415926）

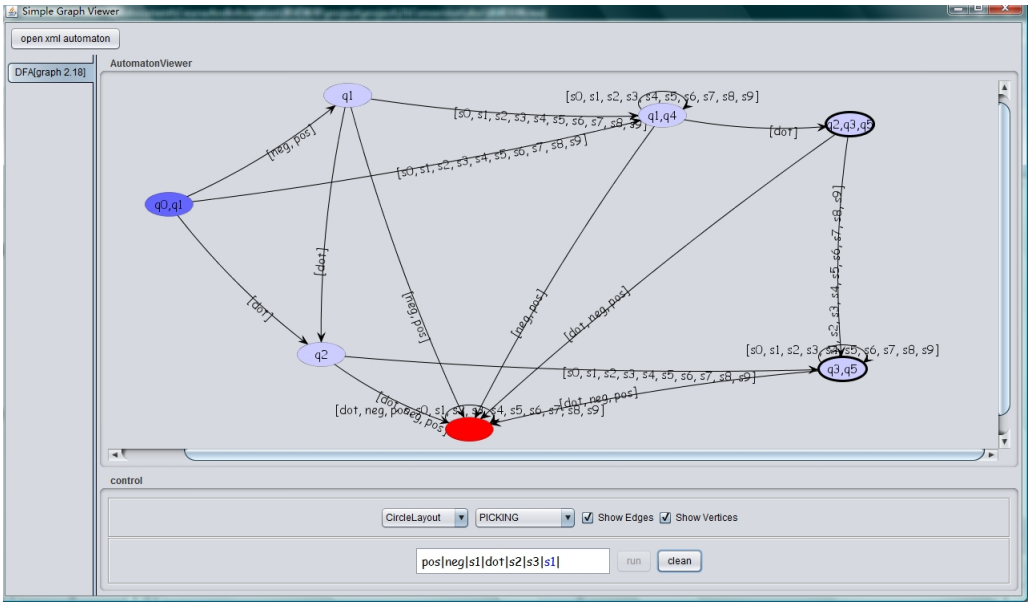

Figure 6: 错误的例子（+-1.231）

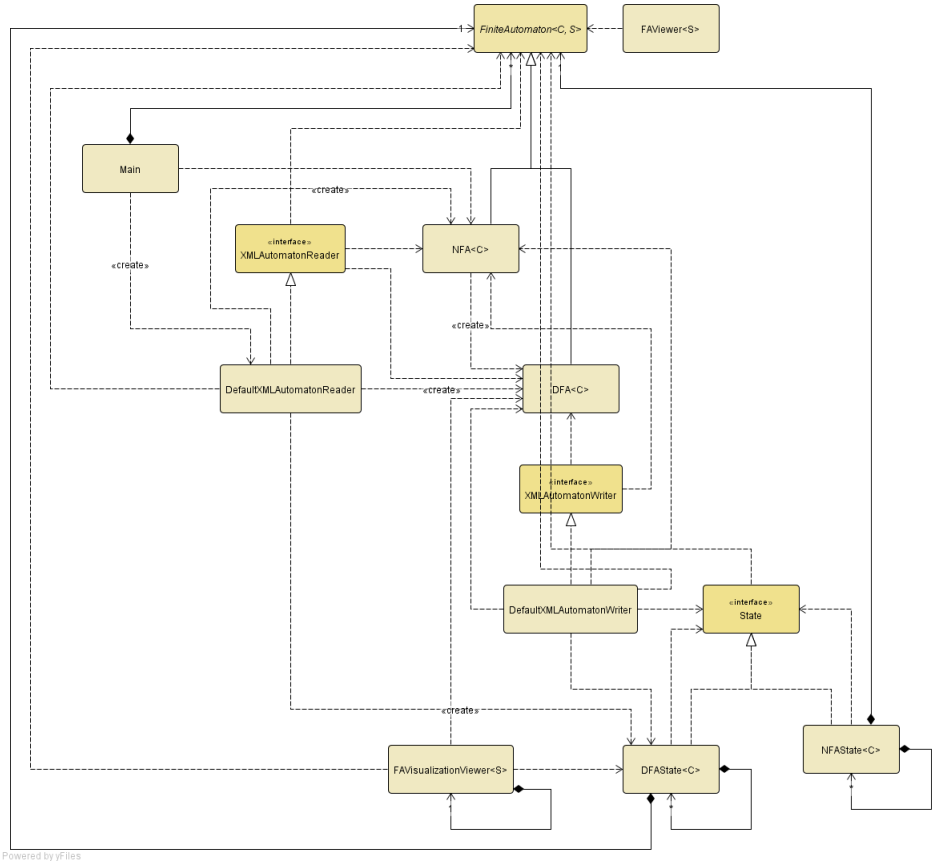

Figure 7: 各个类之间的关系