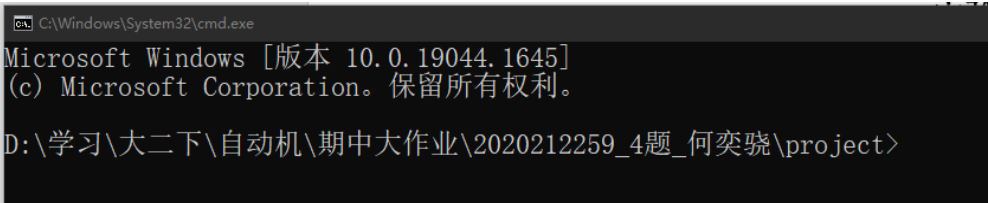


实验报告

课程编号：3132112040 实践课程名称：形式语言与自动机 学年：2021-2022 学期：春

学生姓名	何奕骁	学号	2020212259
指导教师姓名	刘咏彬	起止时间	-
项目名称	RL 模型转换工具 4 号题		
项目内容	<div>一、 实验简介</div> <p>根据学号，我完成的是 4 号题，即将 DFA 转换为最小化 DFA 以及将 DFA 转换为 RG。采用的编程语言为 JavaScript，运行环境为 Node.js（window10 操作系统），编写代码采用的是 vscode。</p> <div>二、 实验环境以及文件介绍</div> <p>project 文件夹中存放的是本次实验的源代码文件以及输入输出的文本文件，其中的 main.js 为本次实验所编写的代码文件，input.txt 文件为输入的 DFA，运行 main.js，将把 input.txt 中输入的 DFA 分别转换为最小化 DFA 以及 RG，并分别写入到 result_MinDFA.txt 以及 result_RG.txt 中。</p> <p>运行本次实验代码 main.js 需提前配置好 Node.js 环境。可以参考这篇文章进行安装：</p> <p>http://t.csdn.cn/0tYlW</p> <p>安装完成后，先按照格式写入 DFA 到 input.txt 中，然后在 main.js 所在目录打开 cmd，如下图：</p> <div></div> <p>输入 node main.js，再按回车键就可以运行了，如下图：</p>		

```

C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19044.1645]
(c) Microsoft Corporation。保留所有权利。

D:\学习\大二下\自动机\期中大作业\2020212259_4题_何奕骁\project>node main.js
读入DFA
{
  q0: { q0: false, q1: false, q2: true, q3: true, q4: true, q5: true },
  q1: { q0: false, q1: false, q2: true, q3: true, q4: true, q5: true },
  q2: { q0: true, q1: true, q2: false, q3: false, q4: false, q5: true },
  q3: { q0: true, q1: true, q2: false, q3: false, q4: false, q5: true },
  q4: { q0: true, q1: true, q2: false, q3: false, q4: false, q5: true },
  q5: { q0: true, q1: true, q2: true, q3: true, q4: true, q5: false }
}

```

运行后，打开 result_MinDFA.txt 与 result_RG.txt，就可以看到将 DFA 依要求转化后的结果了。

三、 实验设计方案

本次实验，我采用了面向对象的方法，抽象出了 DFA 对象，并在 DFA 对象中加入了多个对象属性与对象方法以完成将 DFA 转换为 RG 以及最小化 DFA。

代码的整体结构如下所示：

```

// 导入文件系统模块(fs)
var fs = require("fs");

// DFA对象
var DFA = { ...
}

// 异步读入输入的DFA
fs.readFile('input.txt', function(err, data){
  ...if(err){
    ...return console.error('读取input.txt失败:', err);
    ...}
    ...DFA.creat(data.toString()); // 初始化DFA
    ...DFA.toRG(); // 将DFA转换为RG
    ...DFA.toMinDFA(); // 将DFA转换为最小化DFA
  });
}

```

下面，我将对 DFA 转换为 RG 与 DFA 转换为最小化 DFA 分别进行说明。

DFA 转换为 RG

涉及到的成员属性如下所示：

```

initial_state: '', // 起始状态名称
final_states: {}, // 终止状态集名称集合
states: {}, // 状态集
writeStrRG: '', // 写入文件的Str

```

方法说明：

`creat(data)` 方法：

这是初始化 DFA 的算法，传入的参数 `data` 为读入的 `input.txt`，为字符串类型。该函数先判断 `input.txt` 中输入的 DFA 格式是否正确，若不正确，会报错；若正确，则初始化 DFA 的各成员属性，然后删除掉 DFA 中的不可达状态，完成初始化。

`deleteUnreachableState()` 方法：

这是去除 DFA 中不可达状态的方法，具体代码如下所示：

```
// 去除不可达状态
deleteUnreachableState() {
    .... // 可达状态表
    .... var canVisitTable = {};

    .... // 深度优先算法进行标记
    .... var dfs = (state) => {
    ....     canVisitTable[state] = true;
    ....     if (!canVisitTable[this.states[state]['0']]) dfs(this.states[state]['0']);
    ....     if (!canVisitTable[this.states[state]['1']]) dfs(this.states[state]['1']);
    .... };
    .... dfs(this.initial_state);

    .... // 删除不可达状态
    .... for (var state in this.states) {
    ....     if (!canVisitTable[state]) {
    ....         delete this.states[state];
    ....     }
    .... }
},
```

代码解释：

采用 DFS 算法，从开始状态开始依次访问各状态，若访问到某个状态，就在可达状态表中标记这个状态；最终，未在可达状态表中标记的状态即为不可达状态，删除不可达状态。

`writeRGPrpduction(state)` 方法：

这是拼接 RG 产生式的方法，通过传入 `state` 参数，即指定的状态，拼接出对应 RG 中相应变量的产生式。具体代码如下图所示：

```
// 打印RG的产生式
writeRGPrduction(state) {
    ...var s = state + '->0' + (this.states[state]['0']) + '|1' + (this.states[state]['1']);
    ...if (this.final_states[this.states[state]['0']]) s += '|0';
    ...if (this.final_states[this.states[state]['1']]) s += '|1';
    ...this.writeStrRG += s + '\n';
},
```

toRG() 方法:

这是将 DFA 转换为 RG 的方法，该方法从起始状态开始，调用 writeRGPrduction(state) 方法，得到需要写入到 result_RG.txt 的 RG 的所有产生式，最终，完成 DFA 转换为 RG 并将结果写入到 result_RG.txt 中。

DFA 转换为最小化 DFA

成员属性:

disStateTable: 这是可区分状态表，用来记录所有的状态能否被区分。

associationTable: 这是状态对关联表，用来记录与某个状态对 (q, p) 关联的状态对序列。

minDFAStates: 这是最小化 DFA 的状态集。

方法说明:

minimizationDFA() 方法:

这是 DFA 的极小化算法，通过这个方法，可以构建出 disStateTable，进而可以完成 DFA 的极小化转换。这个方法我参考的是课本上的算法，伪代码如下：

算法 5-1 DFA 的极小化算法。

输入：给定的 DFA。

输出：可区分状态表。

主要数据结构：可区分状态表；状态对的关联表。

主要步骤：

- (1) for $\forall (q, p) \in F \times (Q - F)$ do
 标记可区分状态表中的表项 (q, p) ； /* p 和 q 不可合并 */
- (2) for $\forall (q, p) \in F \times F \cup (Q - F) \times (Q - F)$ 且 $q \neq p$ do
- (3) if $\exists a \in \Sigma$, 可区分状态表中的表项 $(\delta(q, a), \delta(p, a))$ 已被标记 then
 begin
- (4) 标记可区分状态表中的表项 (q, p) ；
- (5) 递归地标记本次被标记的状态对的关联表上的各个状态对在可区分状态表中的对应表项。
- end
- (6) else for $\forall a \in \Sigma$ do
- (7) if $\delta(q, a) \neq \delta(p, a)$ 且 (q, p) 与 $(\delta(q, a), \delta(p, a))$ 不是同一个状态对 then
 将 (q, p) 放在 $(\delta(q, a), \delta(p, a))$ 的关联链表上。

toMinDFA() 方法:

这是将 DFA 转换为最小化 DFA 的方法。该方法先调用 minimizationDFA() 方法得到 disStateTable，然后使用深度优先遍历算法来初始化 minDFAStates，最终调用 writeMinDFA() 方法将转化后的最小化 DFA 写入 result_MinDFA.txt 中。

writeMinDFA() 方法:

这是根据 minDFAStates 将转换后的最小化 DFA 写入到 result_MinDFA.txt 的方法。

四、 测试用例

1. 测试输入不合法内容

输入的 input.txt 如下所示，可见，输入的 DFA 没有起始状态:

```
1  0 1
2  q0 q1 q2
3  q1 q0 q3
4  *q2 q4 q5
5  *q3 q4 q5
6  *q4 q4 q5
7  q5 q5 q5
```

运行 main.js，如下:

```
PS D:\学习\大二下\自动机\期中大作业\2020212259_4题_何奕骅\project> node "d:\学习\大二下\自动机\期中大作业\2020212259_4题_何奕骅\project\main.js:108
读入DFA
输入格式有误: 没有起始状态
d:\学习\大二下\自动机\期中大作业\2020212259_4题_何奕骅\project\main.js:108
    var s = state + '->0' + (this.states[state]['0']) + '|1' + (this.sta
```

打印出了错误信息，没有起始状态。

2. 课本 P153 页示例

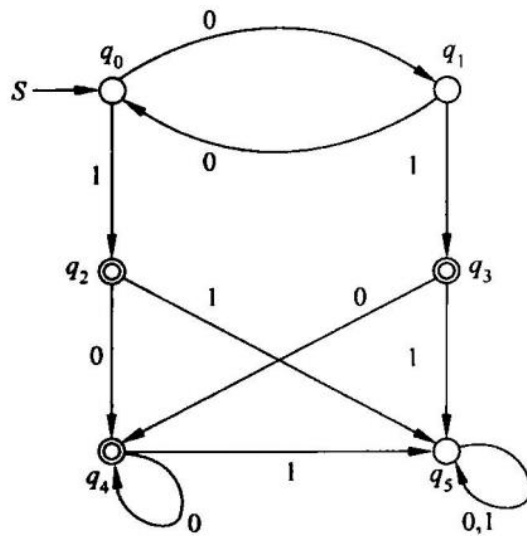
输入的 input.txt 如下所示:

```

0 · 1
#q0 · q1 · q2
q1 · q0 · q3
*q2 · q4 · q5
*q3 · q4 · q5
*q4 · q4 · q5
q5 · q5 · q5

```

可视化:



运行代码，result_RG.txt 如下所示：

```

q0->0q1|1q2|1
q1->0q0|1q3|1
q2->0q4|1q5|0
q3->0q4|1q5|0
q4->0q4|1q5|0
q5->0q5|1q5

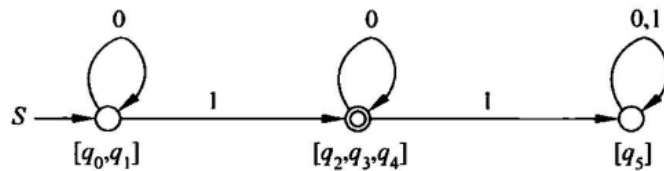
```

可以验证，正确的完成了将 DFA 转换为 RG。

result_MinDFA.txt 如下所示：

```
#[q0,q1] · [q0,q1] · [q2,q3,q4]
*[q2,q3,q4] · [q2,q3,q4] · [q5]
[q5] · [q5] · [q5]
```

可视化:



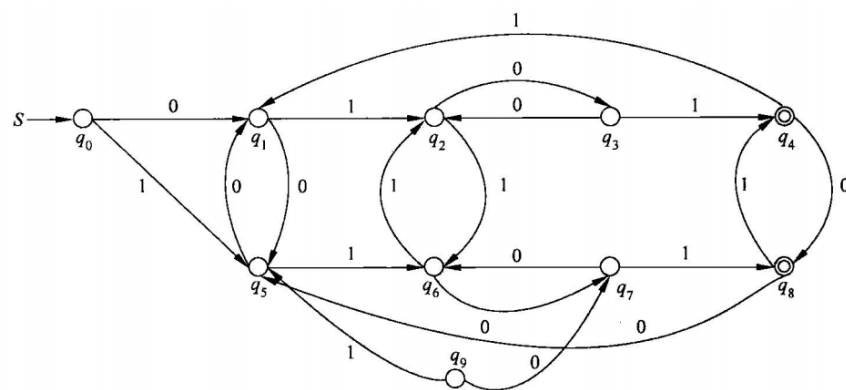
即，正确的完成了将 DFA 转换为最小化 DFA。

3. 课本 P154 页示例

输入的 input.txt 如下所示:

```
0 1
#q0 q1 q5
q1 q5 q2
q2 q3 q6
q3 q2 q4
*q4 q8 q1
q5 q1 q6
q6 q7 q2
q7 q6 q8
*q8 q5 q4
q9 q7 q5
```

可视化:



运行代码，result_RG.txt 如下所示：

```
1  q0->0q1|1q5
2  q1->0q5|1q2
3  q2->0q3|1q6
4  q3->0q2|1q4|1
5  q4->0q8|1q1|0
6  q5->0q1|1q6
7  q6->0q7|1q2
8  q7->0q6|1q8|1
9  q8->0q5|1q4|1
```

可见，正确的完成了将 DFA 转换为 RG（默认去除掉不可达状态）。

result_MinDFA.txt 如下所示：

```
#[q0] · [q1] · [q5]
[q1] · [q5] · [q2]
[q2] · [q3] · [q6]
[q3] · [q2] · [q4]
*[q4] · [q8] · [q1]
[q5] · [q1] · [q6]
[q6] · [q7] · [q2]
[q7] · [q6] · [q8]
*[q8] · [q5] · [q4]
```

参考课本 P156 页，可知正确的完成了从 DFA 到最小化 DFA 的转换。

结 论	<p>通过这次实验，我完成了 DFA 转换为最小化 DFA 以及 DFA 转换为 RG 的任务，在实验过程中我巩固了形式语言与自动机理论课上所学的知识，并增强了我对 DFA，RG 与最小化 DFA 等概念的理解，提升了我的编程能力以及数据结构与算法基础，也提升了我处理 BUG 的能力。总之，这次期中大作业，令我受益匪浅。</p>
--------	---