类 FiniteAutomata 文档

胡双朴
西安电子科技大学 2015 级本科生

版本：0.1
更新：2019.05.29

摘　　要

类 FiniteAutomata 用来在 TCT 工具的 ADS 文件和 FIRE engine 的类 DFA 之间相互转换。

关键词： FIRE engine，TCT，ADS, DFA

目录

1 介绍

类 FiniteAutomata 构成如下

表 1: 类 DFA

Class FiniteAutomata			
名称	类型	属性	说明
Trans	std::vector<Transition>	private	转移关系
F	std::vector<state>	private	结束状态集
Q	std::vector<state>	private	StatePool 预留
V	std::vector<label>	private	输入字符预留
theDFA	std::string	private	预留
num_state	size_t	private	状态数
quite	bool	public	预留
FiniteAutomata()		public	构造函数
FiniteAutomata(std::string str)		public	构造函数
FiniteAutomata(DFA &dfa)		public	构造函数
operator»(std::istream& input, FiniteAutomata& D)		public	输入运算符重载
operator«(std::ostream& output, FiniteAutomata& D)		public	输出运算符重载
operator==(FiniteAutomata& D)	bool		判断是否相等
reconstruct(std::string str)	FiniteAutomata&	public	通过 string 刷新数据
size()	size_t	public	this->size()=num_state
getDFA()	DFA_components	public	用于构造 DFA
adsToDFA(std::string adsfilename)	bool	public	解析 ADS 文件
perform()	bool	public	输出默认的 FA.ADS
perform(std::string filepath)	bool	public	输出到指定的文件
perform(DFA &dfa, std::string filepath)	bool	public	输出指定的 DFA 到指定的文件
clear()	FiniteAutomata&	public	
analyze(std::string& str)	bool	private	解析 DFA 的输出
check(const state& t)	bool	private	检查一个状态是否合法

注. “预留”内容为设计时添加，但是实际使用中未使用的内容，但是也保留下来，留待添加更多功能时使用。`state` 和 `label` 均为 int 型变量。因为 DFA 要求 label 为 char 变量或者 CharRange 变量，为了方便，这里不使用 CharRange 变量，而使用 char 变量，但是 TCT 工具要求 label 为整数，综合因素之下，限定了 label 的范围为 [0,9] 的整数。因为 TCT 工具默认开始状态为“0”，所以要求进行互相转换的 DFA 的开始状态也为“0”。

在进行 FIRE engine 的最小化算法的测试过程中，FIRE engine 的输出格式不便于观察，可以将 FIRE engine 的输出转换成 TCT 工具能识别的 ADS 文件，使用 ADS 文件绘图后方便观察。

FIRE engine 用下面的方式表示一个 DFA

Listing 1: 图1中自动机在 FIRE engine 中的表现形式

```
DFA
Q = [0,4)
S = { 0 }
F = { 3 }
Transitions =
0->{ '0'->1  '1'->0 }
1->{ '0'->2  '1'->0 }
2->{ '0'->3  '1'->0 }
3->{ '0'->3  '1'->0 }

current = -1
```

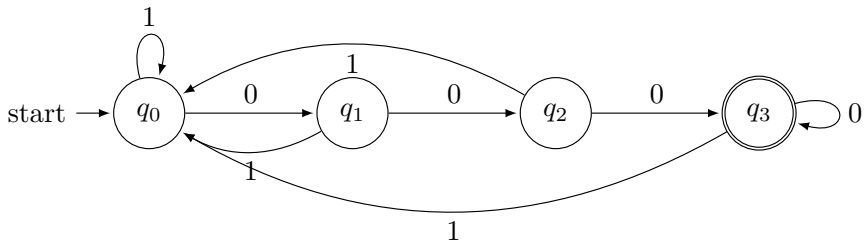

图 1: DFA 示例

2 构造函数

2.1 默认构造函数

默认构造函数 `FiniteAutomata()`

`FiniteAutomata ffa;` 将产生一个空的对象 `ffa`。可以使用 `std::cin>>ffa;` 的方式来从控制台输入。

2.2 通过 DFA 的输出来构造

通过 DFA 的输出来构造 `FiniteAutomata(std::string str)`

下面的例子：

```
DFA dfa1;
……
std::stringstream ss;
```

```
ss << dfa1;
std::string temp=ss.str();
FiniteAutomata ffa(temp);
// to do something
```

2.2.1 刷新数据

reconstruct(std::string str)

此函数用来通过 DFA 的输出刷新当前的对象。

```
DFA dfa1;
……
std::stringstream ss;
ss << dfa1;
std::string temp=ss.str();
FiniteAutomata ffa(temp);
// to do something
// 若ffa 被改变或者需要把它改变成另外一个 DFA
ffa.reconstruct(temp);
```

2.3 通过 DFA 来构造

FiniteAutomata(DFA &dfa)

样例如下

```
DFA dfa1;
……
FiniteAutomata ffa(dfa1);
// to do something
```

3 运算符重载

输入运算符重载：此功能简化了 TCT 工具的输入过程；

输出运算符重载，`std::cout<<ffa<<std::endl;` 将直接把 ffa 转换成 ADS 文件的格式输出到控制台。

== 运算符重载：此功能可用于简单判断 DFA 是否相等。

以下面的例子来说明：

Listing 2: 例子 1

```
DFA
Q = [0,4)
```

```
S = { 0 }
F = { 3 }
Transitions =
0->{ '0'->1  '1'->0 }
1->{ '0'->2  '1'->0 }
2->{ '0'->3  '1'->0 }
3->{ '0'->3  '1'->0 }

current = -1
```

Listing 3: 例子 2

```
DFA
Q = [0,4)
S = { 0 }
F = { 3 }
Transitions =
0->{ '0'->1  '1'->0 }
1->{ '1'->0  '0'->2 }
2->{ '0'->3  '1'->0 }
3->{ '0'->3  '1'->0 }

current = -1
```

可以看到第二条转移关系`1->{ '0'->2  '1'->0 }`和`1->{ '1'->0  '0'->2 }`并不一样，此时若仅仅把输出当作“字符串”来判断，那么将会出错。实际上只是调换了位置而已。本功能可以对状态名和转移关系相同的等价的 DFA 进行简单的判断。（注意：只是简单的判断，DFA 的等价需要从它们接受的 $\mathcal{L}$ 来判断）。

判断两个 DFA 是否等价的例子如下：

Listing 4: 例子 3

```
DFA dfa1;
……
DFA dfa2;
……
FiniteAutomata ffa1(dfa1);
FiniteAutomata ffa2(dfa2);
std::cout<<(ffa1==ffa2)<<std::endl; // dfa1=dfa2?
```

4 如何生成 DFA

4.1 从键盘键入

生成 DFA 主要通过成员函数 `getDFA()`，此函数返回一个 DFA_components 变量，用于实例化类 DFA。例如：

```
FiniteAutomata ffa;
cin>>ffa;
DFA_components dfa_com= ffa.getDFA();
DFA dfa1(dfa_com);
// to do something
```

注. 执行此函数的前提是 FiniteAutomata 对象已经有完整的数据。

4.2 从 ADS 文件生成 DFA

此功能需要先执行成员函数 `adsToDFA()`。传入参数为 ADS 文件的全称。然后执行成员函数 `getDFA()`。

5 如何生成 ADS 文件

5.1 从键盘键入

```
FiniteAutomata ffa;
cin>>ffa;
ffa.perform(); //输出到默认的文件 FA.ADS
// 或者输出到指定的文件 FILE.ADS
ffa.perform("FILE.ADS"); //
```

5.2 用 DFA 生成

用 DFA 构造一个 FiniteAutomata 的方法，见第 1 节——构造函数。得到 FiniteAutomata 对象之后，执行`perform()` 或者`perform(std::string filepath)` 都可以生成 ADS 文件。

另一种方法：

```
DFA dfa1;
……
FiniteAutomata ffa;
ffa.perform(dfa1."FILE.ADS");
```