

**杭州电子科技大学**

**《编译原理课程实践》**

**实验报告**

题 目： 正规表达式转最小化DFA

学 院： 卓越学院

专 业： 计算机科学与技术

班 级： 23186211

学 号： 23050817

姓 名： 吴锴

完成日期： 2024-10-30

正规表达式转NFA

1. **实验目的**

1. 掌握正规表达式与有限⾃动机的基本概念和转换⽅法。

2. 了解⾮确定有限⾃动机（NFA）的构建过程。

3. 熟悉编程实现正规表达式到NFA转换的算法。

4. 提⾼编程能⼒和算法设计的技能。

1. **实验内容与实验要求**

1. 理论背景：正规表达式是⼀种⽤于描述词法单元的形式化表示法，⽽NFA是⼀种⽤于词法分析的状态机。正规 表达式可以通过算法转化为NFA，从⽽实现对字符串的模式匹配。

2. 任务描述：实现正规表达式到NFA的转换算法，并验证⽣成的NFA对给定输⼊字符串的接受性。同时，设计适 合NFA的数据结构，为后续NFA转DFA、DFA最⼩化等实验任务提供基础⽀持。

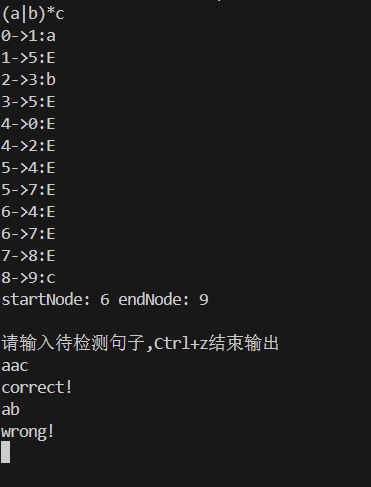
3. \*\* 实验步骤\*\*： 解析输⼊的正规表达式。 构建对应的NFA，包括处理基本符号、连接、并联（或操作）、闭包（星号操作）等运算。 设计并实现合理的数据结构表示NFA，如状态集合、转移关系、初始状态和接受状态。 对NFA进⾏模拟，验证其是否接受给定的输⼊字符串。

4. 案例分析：给定⼀个简单的正规表达式（如a(b|c)\* ），⼿动推导其NFA，并⽤程序实现⾃动⽣成NFA的过程。

1. **设计方案与算法描述**

首先将输入的正则表达式字符之间通过“^”连接，随后转变为后缀表达式，使用Thompson算法将其分解为基本操作，之后对每个符号做不同的处理，实现NFA之间的拼接、并联、闭包操作，逐步合并完成NFA的构建。

1. **测试结果**



1. **源代码**

见附录文件中的REX2NFA

NFA转DFA

1. **实验目的**

1. 掌握⾮确定有限⾃动机（NFA）与确定有限⾃动机（DFA）的基本概念及其转换⽅法。

2. 了解NFA到DFA转换过程中的⼦集构造算法。

3. 实现NFA到DFA的转换算法，并验证DFA的正确性。

4. 设计合理的数据结构，延续上⼀次实验的结构，以便为后续DFA最⼩化实验任务做好准备。

5. 提⾼编程能⼒及算法设计和优化的技能。

1. **实验内容与实验要求**

实验内容

1. 理论背景：NFA是⼀种可以处理多条路径的状态机，⽽DFA是其确定版本，不存在多条路径。通过⼦集构造算 法（Subset Construction），可以将NFA转换为等价的DFA，从⽽实现字符串匹配的确定性处理。

2. 任务描述：实现将NFA转换为DFA的算法，并对转换后的DFA进⾏验证。同时，设计适合DFA的数据结构，使 其兼容前⼀次实验的NFA数据结构。

3. 实验步骤 ： 理解⼦集构造算法的原理，包括ε-闭包的计算和状态集合的映射。 利⽤⼦集构造算法，将NFA转换为DFA。 设计并实现DFA的数据结构，确保其能够表示状态集合、状态转换、初始状态和接受状态。 验证DFA的正确性，对⽐DFA与NFA在同⼀组测试输⼊上的匹配结果。

实验要求：

1.输⼊输出要求

输⼊：⼀个NFA（包括状态集、转换表、初始状态和接受状态集合）和多个测试字符串。 输出：⽣成的DFA状态集合及其转换关系，指明每个测试字符串是否被DFA接受。

2.算法要求

实现⼦集构造算法，将NFA状态集合的⼦集映射为DFA的单个状态。 处理ε-闭包及其状态转换，⽣成对应的DFA。

3.数据结构要求

在上⼀实验的基础上，设计DFA的数据结构，包含状态集合、转换关系、初始状态和接受状态集合的表 示。 确保数据结构可以⽀持后续的DFA最⼩化任务，便于后续实验任务的延续。

4.程序要求

使⽤C/C++、Java、Python等语⾔编写程序，代码结构清晰，具备良好的注释。 提供详细的实验报告，包括算法设计、实现过程、测试结果和问题分析。

5.实验报告要求

【整合到最后提交的个⼈所有实验报告中，加上⽬录】 描述实验⽬的和内容。 解释⼦集构造算法的原理、步骤和数据结构的设计思路。 给出测试⽤例和结果，分析测试数据的正确性。 总结实验的收获和遇到的挑战。

1. **设计方案与算法描述**

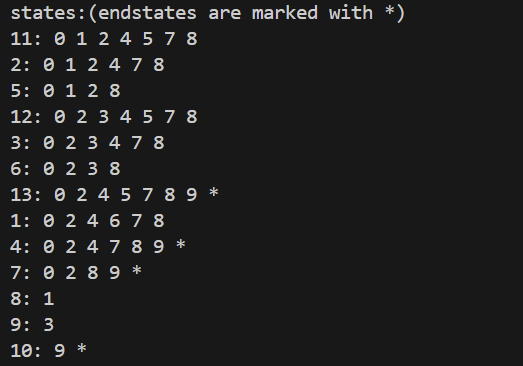
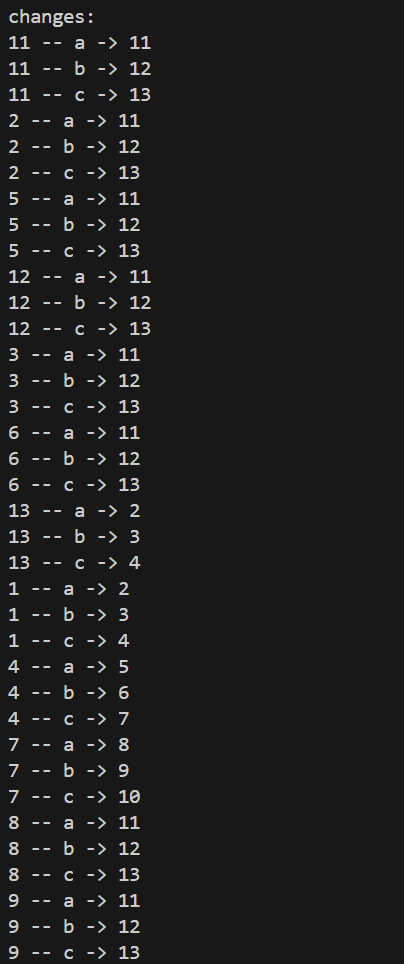
由于状态的唯一性，故而采用c++标准库的set数据类型，用于存储每个状态中的节点，继承之前NFA的设计思路，采用c++标准库的map来保存状态之间的转换，也便于之后最小化DFA时对其进行查找。

算法上模拟实现子集构造法，将NFA状态集合的子集映射为DFA的单个状态，处理ε-closure闭包及其状态转换，生成对应的DFA。

1. **测试结果**

使用上节实验课设计的REX2NFA，将(a|b)\*c转换为NFA后，将NFA输入程序，通过本次课设计的NFA2DFA程序，得到的NFA如下：

1.输入的NFA： 2.得到的结果：

(a|b)\*c

6 9

0 1 a

1 5 E

2 3 b

3 5 E

4 0 E

4 2 E

5 4 E

5 7 E

6 4 E

6 7 E

7 8 E

8 9 c

1. **源代码**

见附录文件夹 NFA2DFA

DFA转最小化DFA

1. **实验目的**

1. 掌握确定有限⾃动机（DFA）的最⼩化原理和算法，尤其是Hopcroft算法（即课上所讲的“求异法”）。

2. 学习DFA状态等价性的判定⽅法，理解最⼩化过程中的分割和合并策略。

3. 实现DFA最⼩化算法，并验证最⼩化DFA的正确性。

4. 延续前两次实验的设计，确保数据结构能贯通整个⾃动机系列实验。

5. 提⾼算法优化和编程实现能⼒，增强对编译原理的理解。

1. **实验内容与实验要求**

实验内容

1. 理论背景：DFA最⼩化是将DFA状态数减少到最⼩的过程，通过合并等价状态，实现最优的状态机表示。

Hopcroft算法是求异法的⼀种⾼效实现，它通过维护状态的分割并使⽤快速查找机制来优化最⼩化过程。

2. 任务描述：实现DFA最⼩化算法，将给定的DFA简化为状态数最少的等价DFA。验证最⼩化DFA的正确性，并

对⽐最⼩化前后的状态数量。

3. 实验步骤

理解Hopcroft算法的基本原理，包括状态等价的判定标准和状态合并的⽅法。

实现Hopcroft算法，将原DFA简化为等价的最⼩化DFA。

设计合理的数据结构表示最⼩化后的DFA，确保其与前两次实验的NFA和DFA数据结构保持⼀致。

验证最⼩化DFA的正确性，确保其接受的语⾔与原DFA相同。

实验要求

1. 输⼊输出要求

输⼊：⼀个DFA（包括状态集合、状态转换表、初始状态和接受状态集合）。

输出：最⼩化后的DFA状态集合及其转换关系，指明最⼩化前后的状态数和状态转换关系。

2. 算法要求

实现Hopcroft算法，通过分割状态集合和快速查找机制来最⼩化DFA。

⽀持状态等价性判定及状态的合并操作。

3. 数据结构要求

设计适合Hopcroft算法的⾼效数据结构，如⽤于记录状态分割的集合、合并后的状态转换表等。

保持与前两次实验的数据结构⼀致，⽅便整个⾃动机系列实验的贯通实现。

4. 程序要求

使⽤C/C++、Java、Python等语⾔编写程序，代码结构清晰，具备良好的注释。

提供详细的实验报告，包括算法设计、实现过程、测试结果和问题分析。

5. 实验报告要求【整合到最后提交的个⼈所有实验报告中，加上⽬录】

描述实验⽬的和内容。

解释Hopcroft算法的原理和实现步骤，说明数据结构的设计思路。

给出测试⽤例和结果，分析最⼩化前后的差异。

总结实验的收获和遇到的挑战。

1. **设计方案与算法描述**

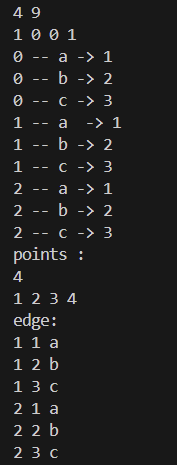
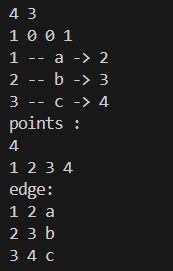
1.读取输⼊DFA，包括状态集合、状态转换表、初始状态和接受状态。

2.根据接受状态和⾮接受状态的划分，初始化状态分割。

3.使⽤Hopcroft算法细化状态分割，按照输⼊符号集逐步分割等价状态，直⾄⽆法再细分。在存储方式上采用经典的并查集算法。

4.输出最⼩化前后的状态数量和转换关系。

1. **测试结果**



1. **源代码**

见附录DFA2MINIDFA