清華大學
Tsinghua University

形式语言与自动机

第一次课程实验

NFA的带路径执行

2024年3月18日

实验概述

- 你了解/使用过正则表达式吗?
  - 例1：批作业脚本，“2022011234_小明_5678.pdf”
  - 例2：小明，13000000000，北京市海淀区清华大学东配楼
  - 例3：（来源于实验所用OJ的真实代码）

```
const operation = /^\s*(status|score)\((\d+)\)\s*(.*)$/m;
```

编程语言中的正则表达式

- **课上讲过的正则表示（正则表达式）**
  - 只允许三种运算：连接·，或+，星闭包*
  - 例：abc，ab+bc，(ab)*c(d+e)
  - 已经证明，这种正则表示具有和NFA等价的表达能力（即能够表示任意的正则语言）
  - 存在问题：语法有些繁琐，不够实用

编程语言中的正则表达式

编程语言中的正则表达式

- 可以认为是对课上讲过的正则表达式的语法扩充
- （其最原始的版本）**表达能力不变（任意正则语言）**
- 在各种编程语言中通用：绝大多数编程语言都有正则表达式相关的库，如：
  - C++：std::regex
  - Python：import re
  - JavaScript：正则表达式字面量/abc/
  - Java：java.util.regex
  - Go：regexp
  - Rust：regex

实验概述

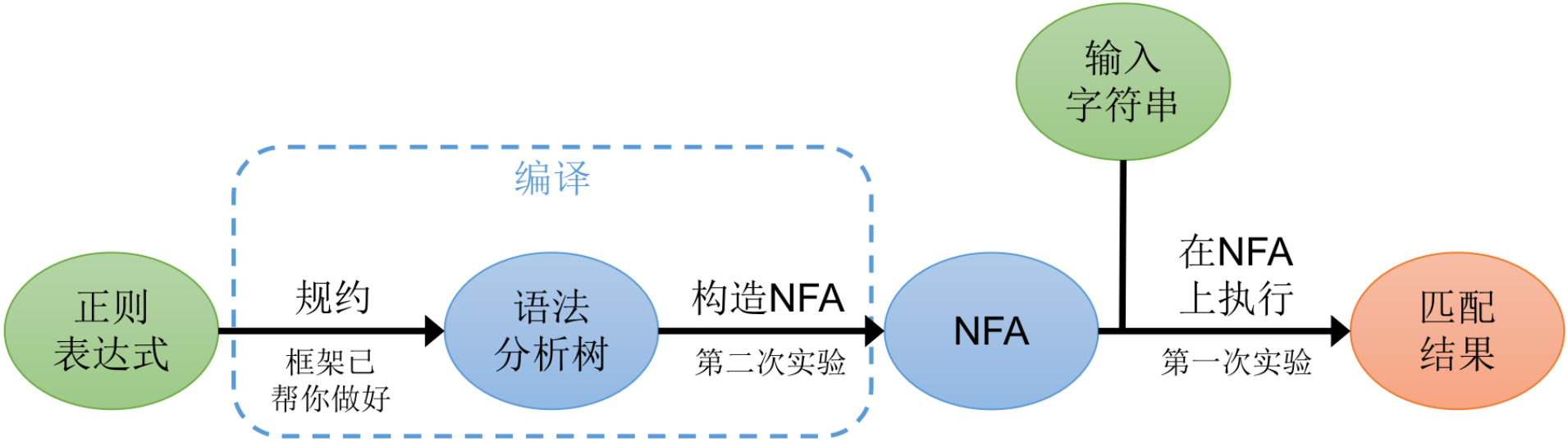
实验目标

- 实现一个正则表达式的执行器
  - 能够对给定的正则表达式和字符串，给出匹配的结果（包含捕获分组的结果）

实验概述

实验安排

- 本课程的实验将分为三个阶段（三次作业）进行，将正则表达式的解析和执行过程进行拆分，并循序渐进
  - **第一次：NFA的带路径执行**
  - 第二次：简单正则表达式的NFA构造
  - 第三次：捕获分组等特性+复杂正则表达式



实验概述

编程方法

- 编程语言：C++ 或 Python
- 编程方法：在给定的代码框架上进行修改
  - 完成相应的函数
- 依赖要求：只能使用C++ stdlib / Python系统库（正则表达式库除外），不能引入其他的外部依赖库
- 提交方法：使用OJ https://oj.starrah.cn/

第一次实验具体说明

- 给定一个NFA和一个字符串
  - 输入的给出方法：（带有标签的、易于人类阅读的）文本文件
  - 代码框架中会从stdin读取输入，并实现简单的解析，生成一个类供你使用/扩展
    - 不同编程语言的数据结构不同，详见具体语言的框架的代码注释
  - NFA包括：状态个数、（初态固定为0）、终态集合、转移列表
  - 状态转移的字符可以包括：ε转移\e，任意ASCII字符（除\0 \r \n外），ASCII字符区间如A-Z，\d \s \w \D \S \W

第一次实验具体说明

- 给定一个NFA和一个字符串
  - NFA包括：状态个数n（状态为0~n-1）、（初态固定为0）、终态集合、转移列表
  - 状态转移的字符可以包括：ε转移\e，任意ASCII字符（除\0 \r \n外），ASCII字符区间如A-Z，\d \s \w \D \S \W
  - 输入字符串：可由除\0 \r \n外的任意ASCII字符组成。
    - 注意非打印字符也是可以包含的

```
states: 4
final: 2 3
rules:
0->1 a b \d
1->2 a-z - _
2->3 \e d 1
input: 1bd
```

状态个数

终态列表

状态转移规则
一个一行

输入字符串

第一次实验具体说明

- 要求输出：
- 拒绝（Reject）
- 接受时，需要给出一条从初态到终态的完整变迁路径
  - 格式：状态 字符 状态 字符 状态...
  - 对下图的例子：应输出0 1 1 b 2 d 3
  - 实际上，输出是通过让函数返回特定的对象完成的（无需手动操作stdout），详见对应编程语言的代码注释

```
states: 4
final: 2 3
rules:
0->1 a b \d
1->2 a-z - _
2->3 \e d 1
input: 1bd
```

状态个数

终态列表

状态转移规则

一个一行

输入字符串

第一次实验具体说明

- 作业要求：
  - 在OJ上提交，全自动化评测，按通过的测例给分
    - 60%公开测例，40%隐藏测例
  - 不需要提交文档
  - 会进行代码抄袭检测，**抄袭者一律0分**

第一次实验具体说明

- 作业要求：
  - 在OJ上提交，全自动化评测，按通过的测例给分
    - 60%公开测例，40%隐藏测例
  - 不需要提交文档
  - 会进行代码抄袭检测，**抄袭者一律0分**

第一次实验具体说明

- OJ使用： https://oj.starrah.cn/ （需要清华校园网环境）

第一次实验具体说明

- OJ使用： https://oj.starrah.cn/ （需要清华校园网环境）

第一次实验具体说明

- OJ使用： https://oj.starrah.cn/ （需要清华校园网环境）

第一次实验具体说明

- OJ使用细则
  - 1. 成绩以最后一次提交为准，即“作业”页面下方“状态”一栏所显示的分数。

  - 2. 关于重测（十成测）：
    - 重测会在DDL约一周后进行，进行后会通知大家，大家此时再登录OJ的作业页面看到的即为最终成绩
    - 在本次重测后补交的同学，应主动联系助教申请重测（仅限一次）；未申请的同学将在期末统一重测。

实验思路

- NFA该如何执行?
  - 课上主要讲解的思路（非回溯法）

扩展转移函数δ^*

- 基础：$\delta^*(q,\varepsilon) = \mathrm{EC}(q)$
- 归纳：$\forall x \in \Sigma^*, a \in \Sigma$(即$a \neq \varepsilon$),

$$\delta^*(q,xa) = \mathrm{EC}\left(\bigcup_{p\in\delta^*(q,x)} \delta(p,a)\right)$$

- 即：对$\delta^*(q,x)$中的每个状态，都可以用δ转移出一个状态集合；将这些状态集合取并集后，对结果中的每个元素求ε-闭包，再并在一起。

实验思路

- NFA该如何执行?
  - 课上主要讲解的思路（非回溯法）
    - 本质是计算NFA的、从初态的、关于输入字符串的扩展转移函数$\delta^*(0,w)$。
    - 始终维护一个“当前可达的状态集合S”，初始化为初态的ε闭包
    - 依次处理每个输入字符：对S中的每个状态，求其在字符a下的转移，并起来再求ε闭包
    - 直到输入完整个字符串，看看S中是否有某个状态是终态

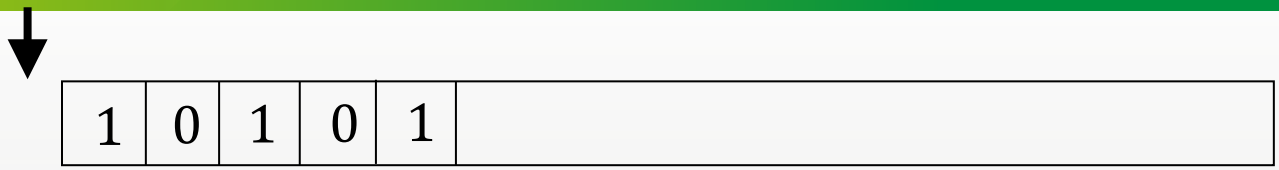
实验思路

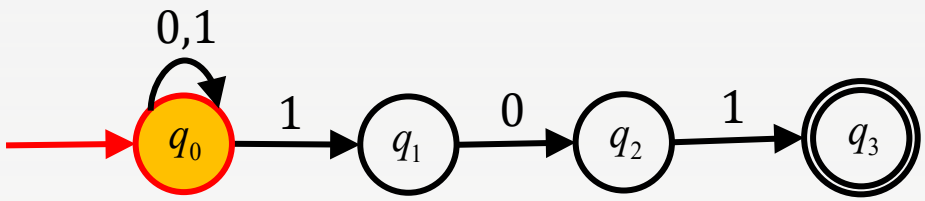
- NFA该如何执行?
  - 课上主要讲解的思路（非回溯法）
  - 问题:
    - **如何给出接受的路径?**
    - 如何处理^、$等情况?
    - 有多条路径时，如何选择给出哪条路径（处理贪婪匹配和非贪婪匹配的区别）？
  - 实际上，我们并不推荐大家用这种方法，而是更推荐大家用下面讲的回溯法

实验思路

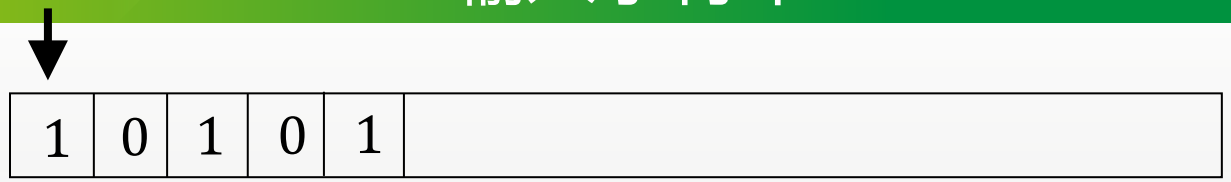
- NFA该如何执行?
  - 回溯法：第三讲开头讲NFA的定义时已有提到
  - 基于深度优先搜索(DFS)，一条一条地试探状态转移的路径，不成就回溯
  - 首先，回忆一下人类执行的过程

输入字符串

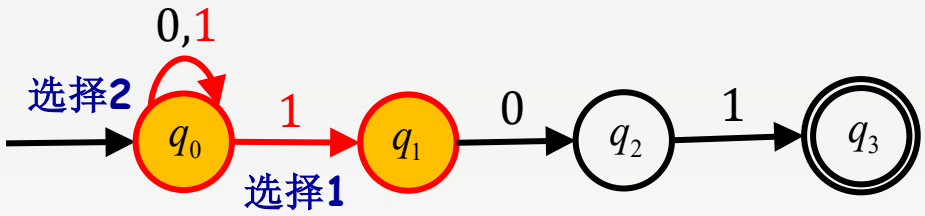




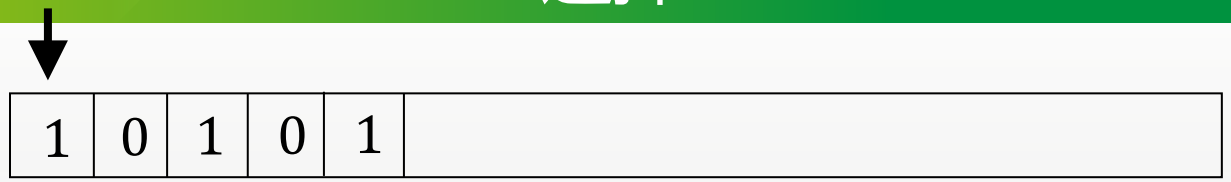
输入字符串



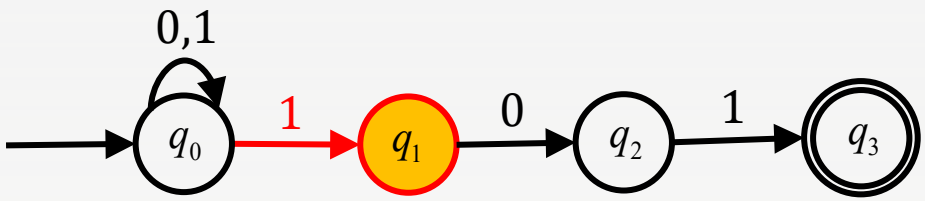
两种选择



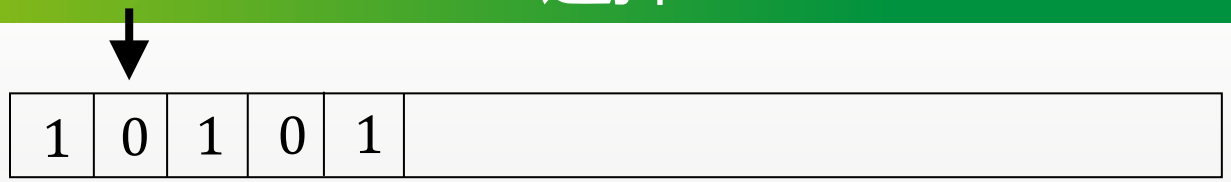
选择1

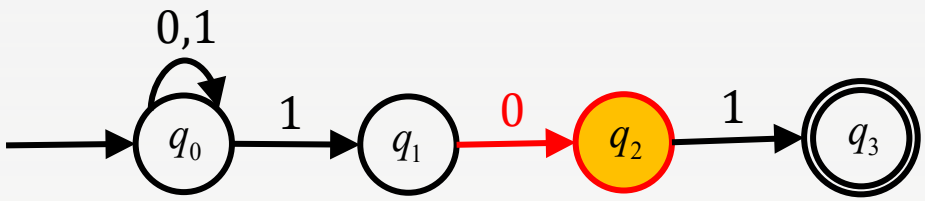


第一种选择

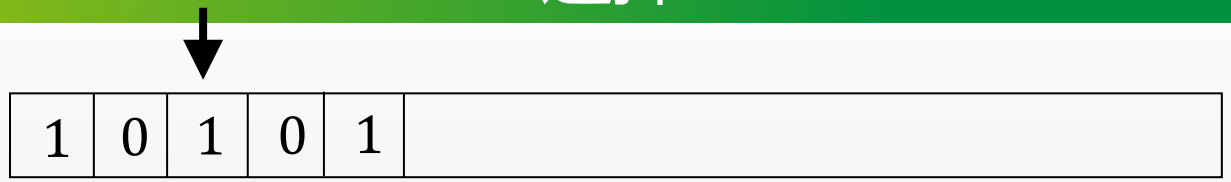


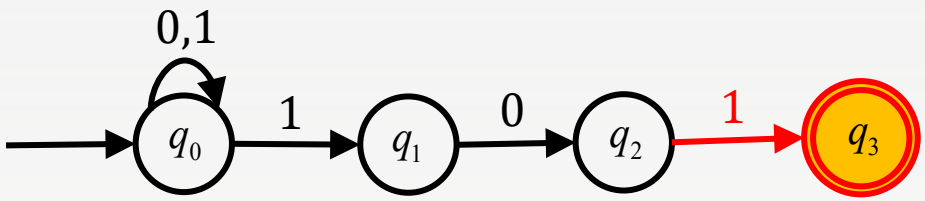
选择1



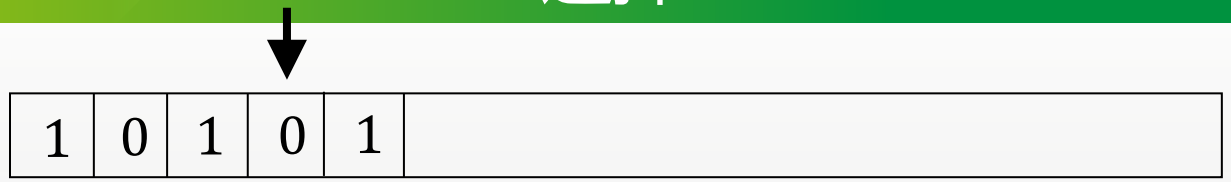


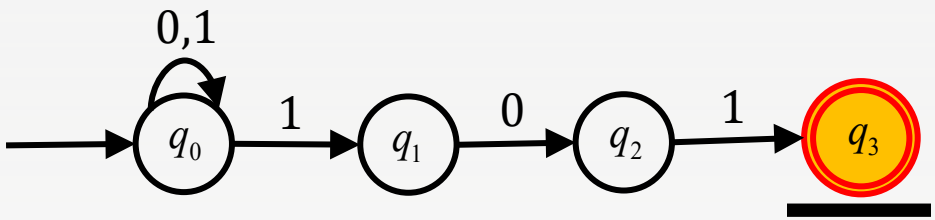
选择1





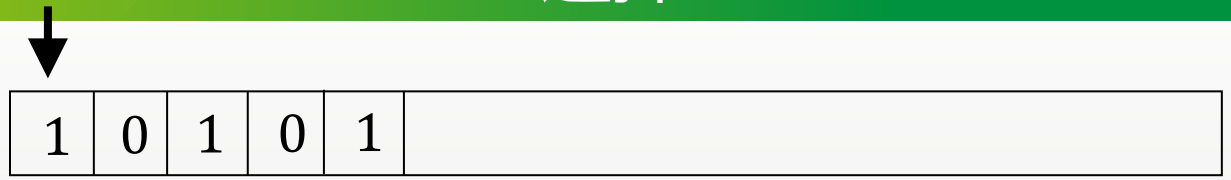
选择1



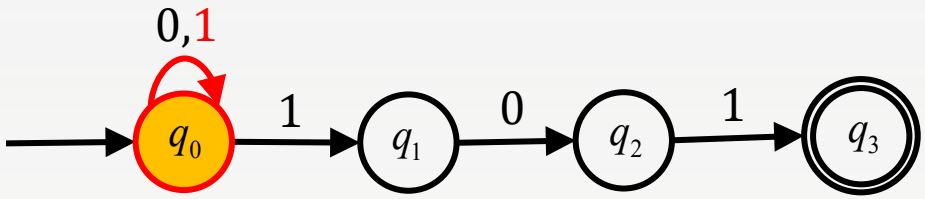


没有迁移
"reject"

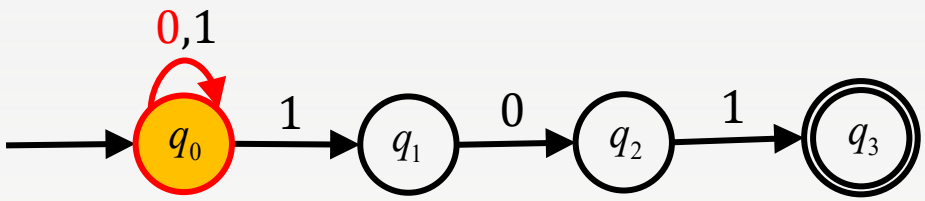
选择2



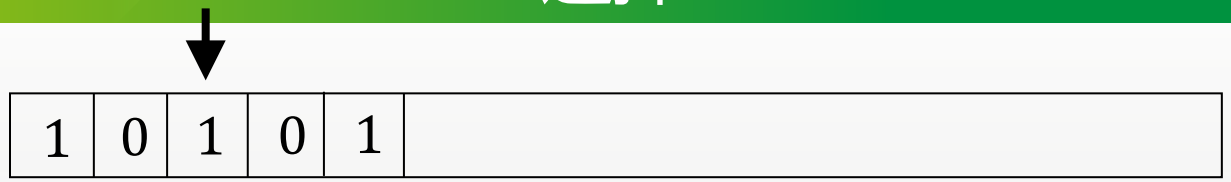
第二种选择



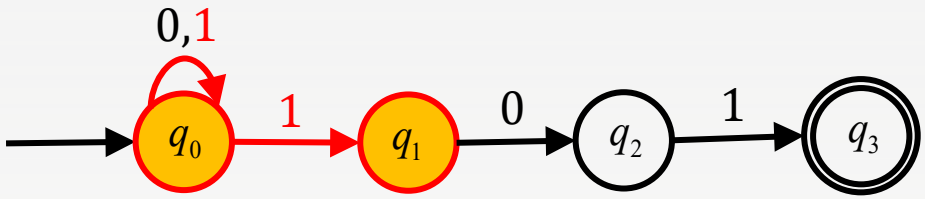
选择2



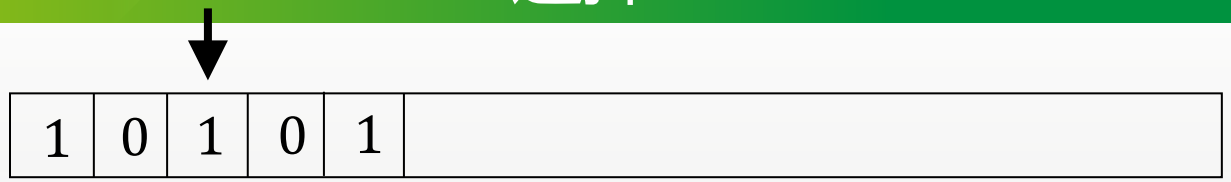
选择2



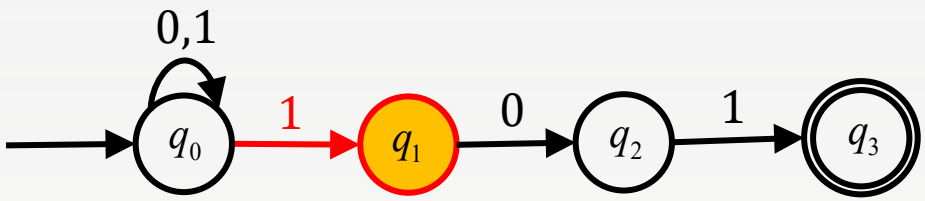
又是两种选择！



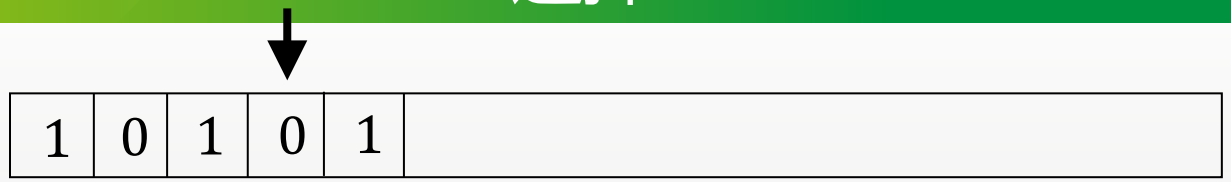
选择2.1

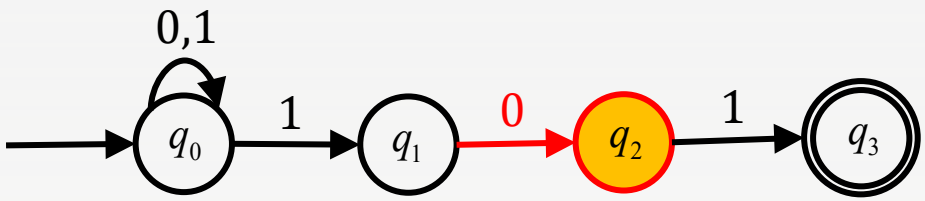


选择2.1

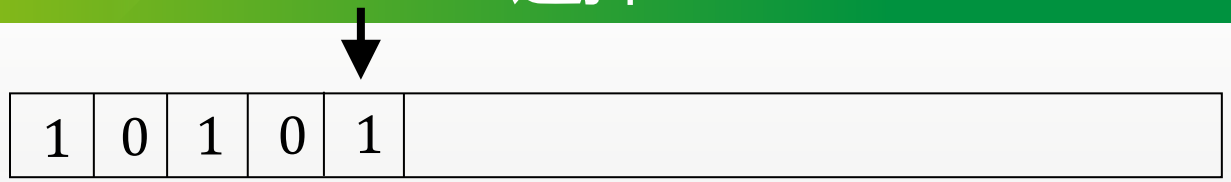


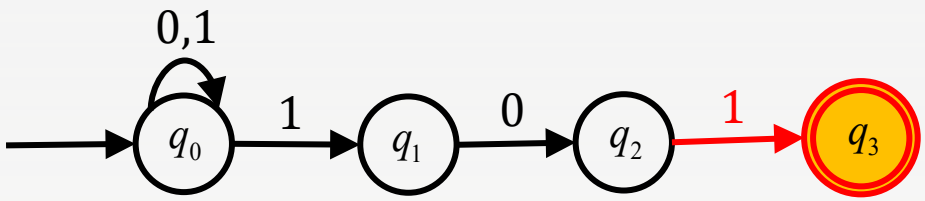
选择2.1



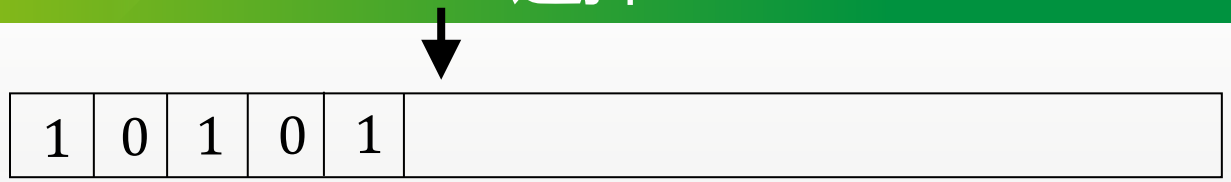


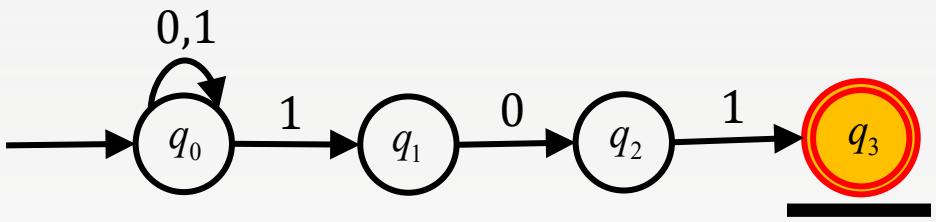
选择2.1





选择2.1

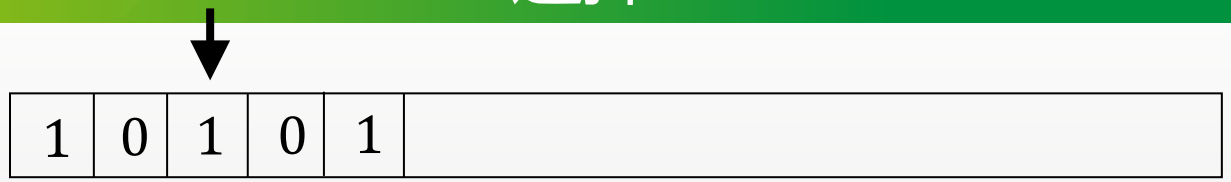




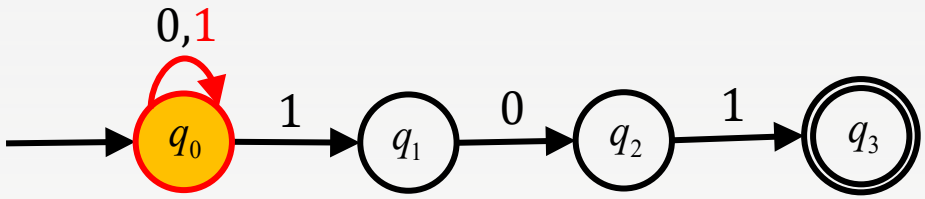
输入串读完后停在终态

accept

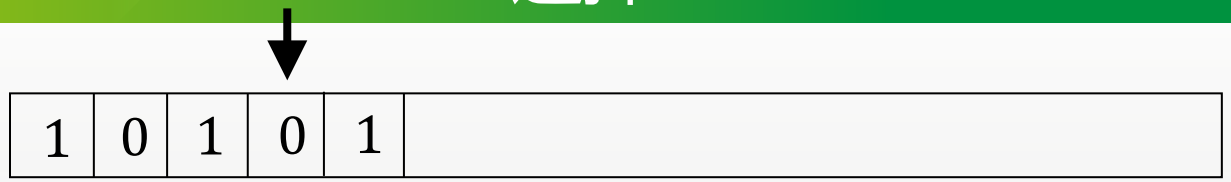
选择2.2

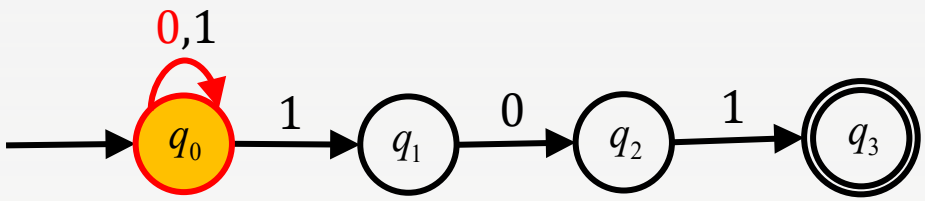


选择2.2

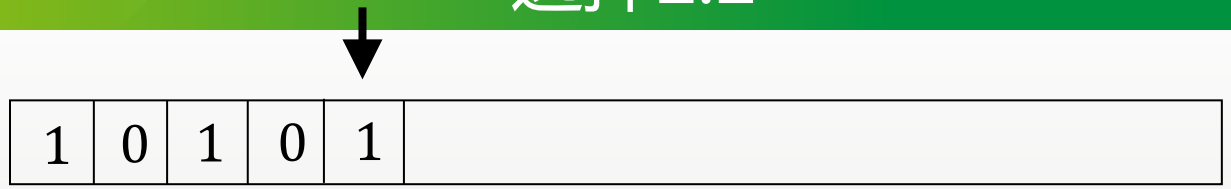


选择2.2

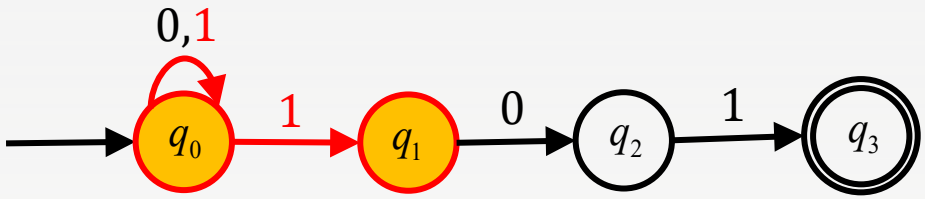




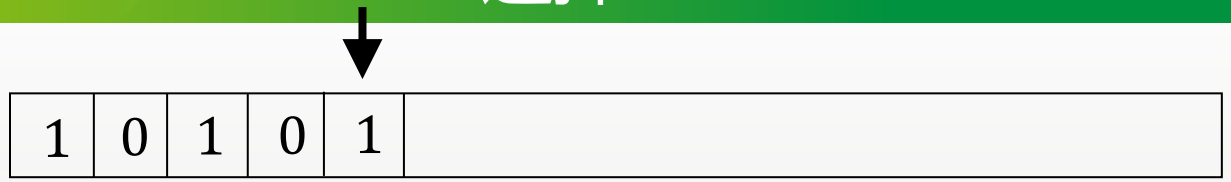
选择2.2



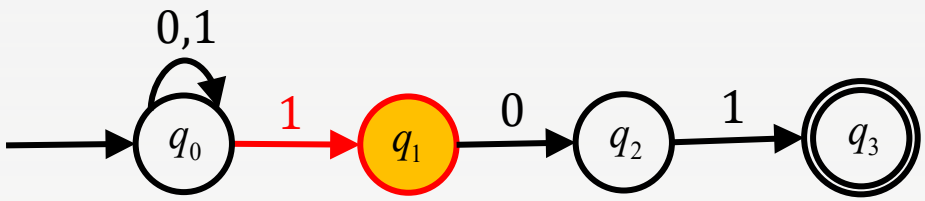
又是两种选择！



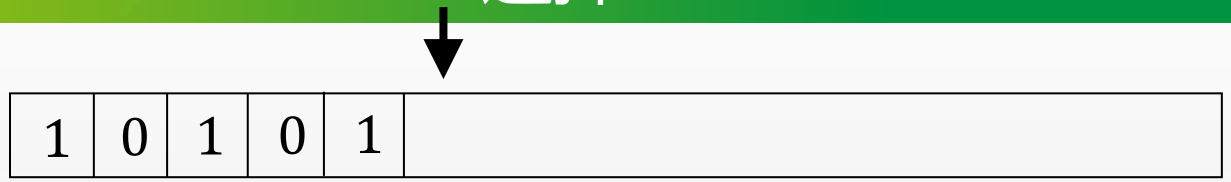
选择2.2.1



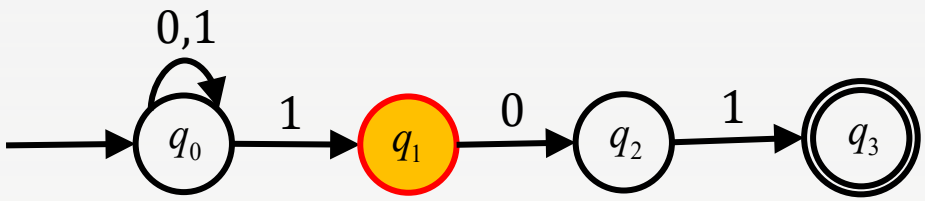
选择2.2.1



选择2.2.1

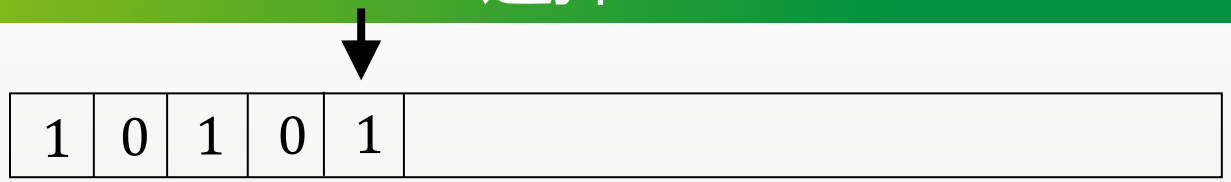


选择2.2.1

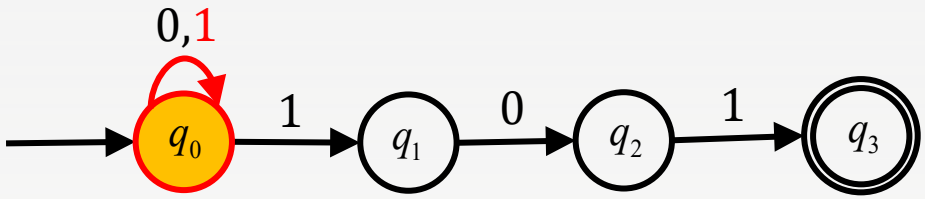


输入串读完后停在非终态
"reject"

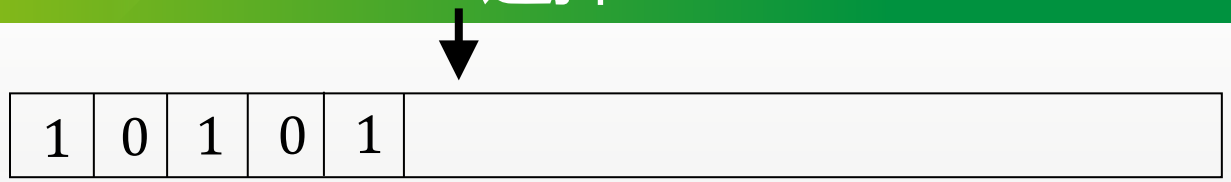
选择2.2.2



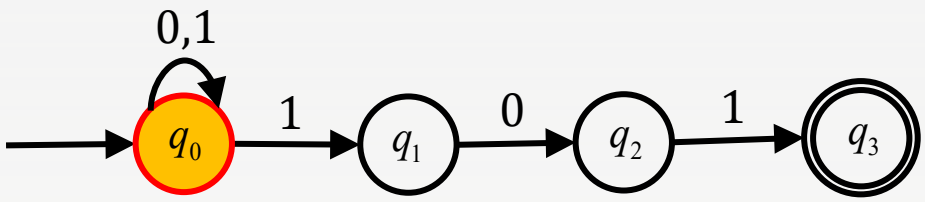
选择2.2.2



选择2.2.2



选择2.2.2



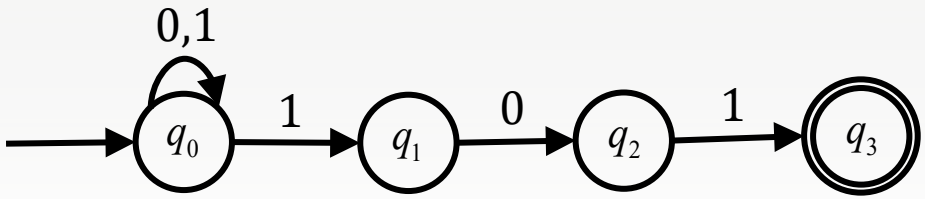
输入串读完后停在非终态
"reject"

总结

1	0	1	0	1	

共四条路径：1X，2.1✓，2.2.1X，2.2.2X
故字符串10101被该自动机接受；

每次在q_0状态、输入字符1，都会分叉出两条路径
思考：上述过程与DFS深度优先搜索的联系



实验思路

- 回溯法：基于DFS，所以使用栈
- 要点提示：
  - 如何处理\d、A-Z等情况?
    - 方法一：分拆成许多状态转移。\d 10个, \w 63个, \D \S更多。（不推荐）
      - 用长为128的数组（起到字典的作用，因为是ASCII）存
    - 方法二：使用一个函数，返回当前位置的转移规则和当前的输入字符是否匹配
      - 下面的伪代码中`rule.match`是这个思路

实验思路

- 回溯法：基于DFS，所以使用栈
- 伪代码

```
stack=[(0, input)]//(状态，剩余输入串)
while !stack.empty():
  q, str = stack,pop()
  if q是终态 && str是空串:
    return true//找到了
  for rule in rules[q].reverse():
    if rule.by == ε:
      stack.push((rule.dest, input))
    else if rule.match(input[0])://即规则与当前输入字符匹配
      //去掉input的首字母
      stack.push((rule.dest, input[1:]))
```

实验思路

这里的伪代码似乎没办法给出状态转移的路径。该如何修改呢?

```
stack=[(0, input)]//(状态，剩余输入串)
while !stack.empty():
  q, str = stack.pop()
  if q是终态 && str是空串:
    return true//找到了
  for rule in rules[q].reverse():
    if rule.by == ε:
      stack.push((rule.dest, input))
    else if rule.match(input[0])://即规则与当前输入字符匹配
      //去掉input的首字母
      stack.push((rule.dest, input[1:]))
```

实验思路

- 回溯法：基于DFS，所以使用栈
- 伪代码

```
stack=[(0, input, 0)]//(状态，剩余输入串, 当前是第几步)
while !stack.empty():
  q, str, step = stack.pop()
  path[step] = (q, str)//path中记录的是第x步的状态和剩余串
  if q是终态 && str是空串:
    return backtrace(path)//找到了，返回路径
  for rule in rules[q].reverse():
    if rule.by == ε:
      stack.push((rule.dest, input, step+1))
    else if rule.match(input[0])://即规则与当前输入字符匹配
      //去掉input的首字母
      stack.push((rule.dest, input[1:], step+1))
```

实验思路

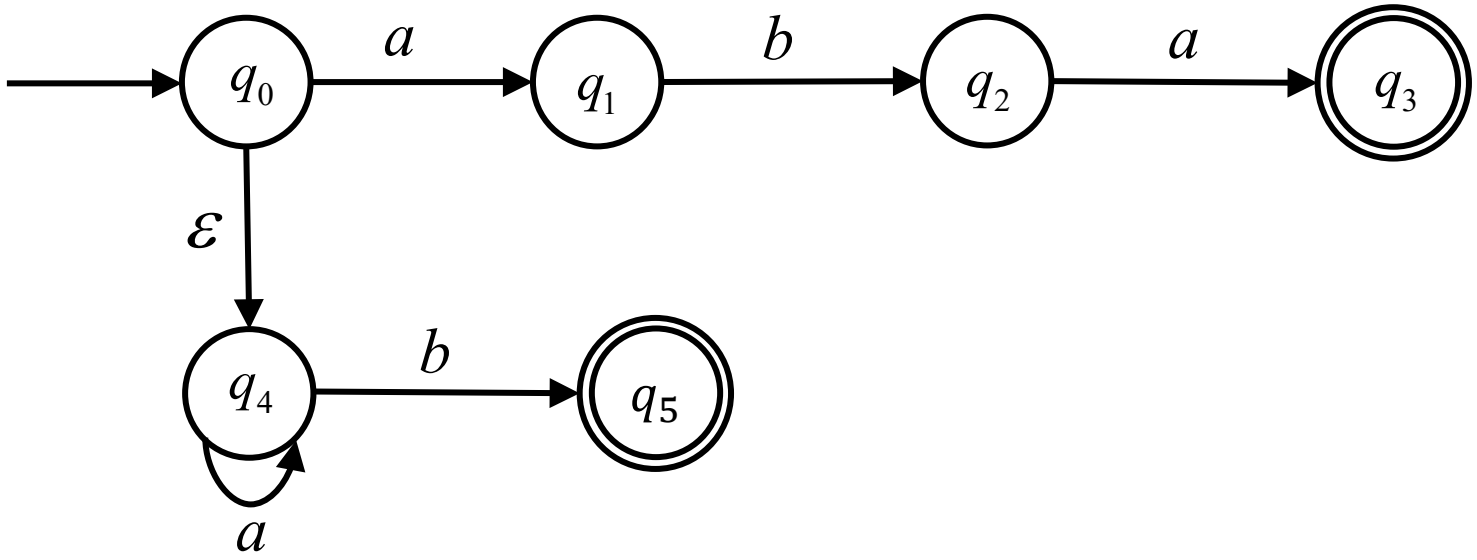
- 回溯法：基于DFS，也可以写成递归的形式。
- 伪代码

```
func run(q, str, step):
  path[step] = (q, str)//path中记录的是第x步的状态和剩余串
  if q是终态 && str是空串:
    result = backtrace(path)
  for rule in rules[q].reverse():
    if rule.by == ε:
      run(rule.dest, input, step+1)
    else if rule.match(input[0])://即规则与当前输入字符匹配
      run((rule.dest, input[1:],step+1))//去掉首字母
    if result != null:
      return//如果已经产生结果，立即退出递归即可
```

- 思考：backtrace函数怎么实现？

实验思路

- 回溯法：基于DFS
- 演示



- 接受串：aab
- 拒绝串：abaa

清華大學
Tsinghua University

谢谢！

欢迎提问！