

# プログラミング応用

<http://bit.ly/kosen02>

Week10@後期(week25 in 2016)  
2016/12/1

- 12/8(木)は米国出張のため休講
- 先週の3の倍数のプログラムは  
講義資料ページにアップしてあります。

# 本日の内容

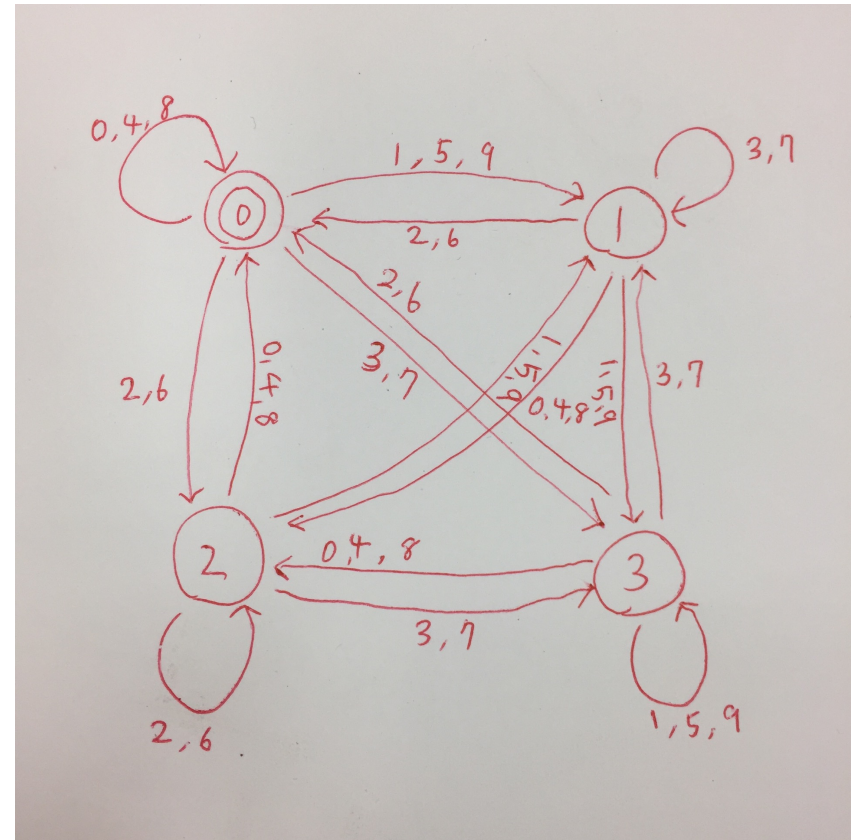
- 講義：
  - 復習：決定性有限オートマトンの概要
  - 非決定性有限オートマトンの表現方法
  - grepコマンドで正規表現を扱う
- 演習：
  - 非決定性有限オートマトン
    - 受理する文字列を判定
    - 状態遷移図 $\leftrightarrow$ 正規表現の相互変換
  - grepコマンドで「吾輩は猫である」を解析

# 本日の内容

- 講義：
  - 復習：決定性有限オートマトンの概要
  - 非決定性有限オートマトンの表現方法
  - grepコマンドで正規表現を扱う
- 演習：
  - 非決定性有限オートマトン
    - 受理する文字列を判定
    - 状態遷移図 $\leftrightarrow$ 正規表現の相互変換
  - grepコマンドで「吾輩は猫である」を解析

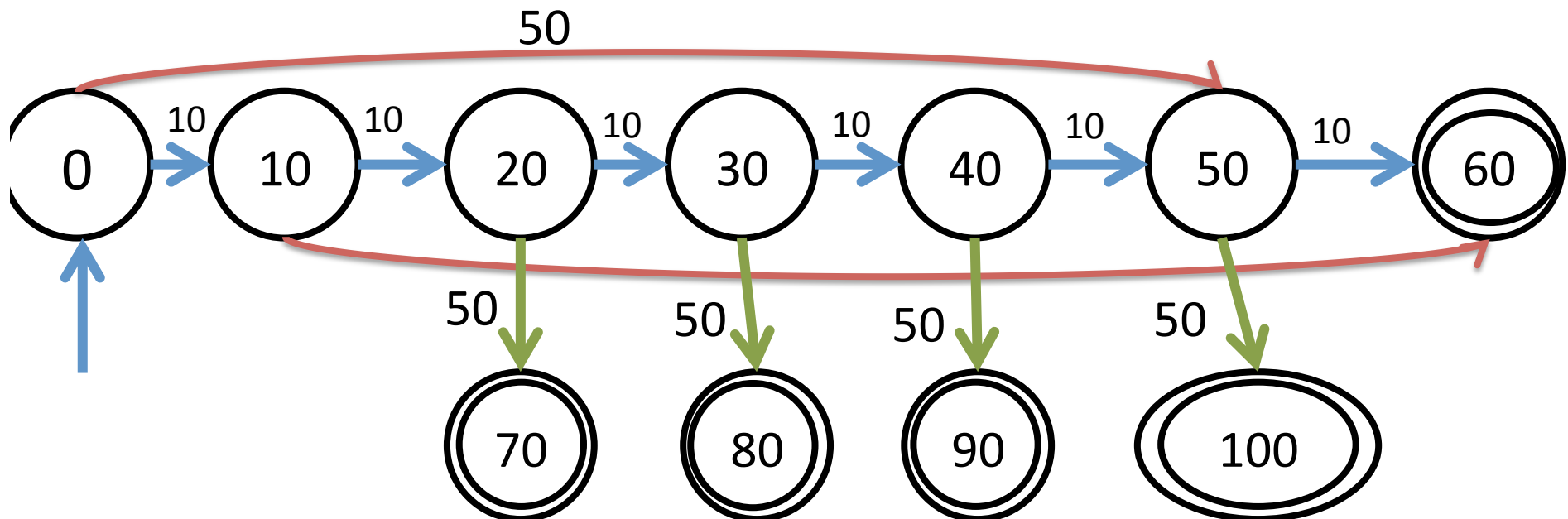
# 先週の演習

- 4の倍数の判定
  - 上位桁から割っていき、  
下二桁が4で割り切れれば4の倍数
  - 条件が多いので実装は不要



# 1. 状態遷移図でオートマトンを表現

- オートマトンの状態と遷移をグラフで表現
  - ノードは状態を表現
    - 初期状態には入り矢印をつける。終了状態は二重丸。
  - 辺: 遷移先を表現



## 2. 形式的定義でオートマトンを表現

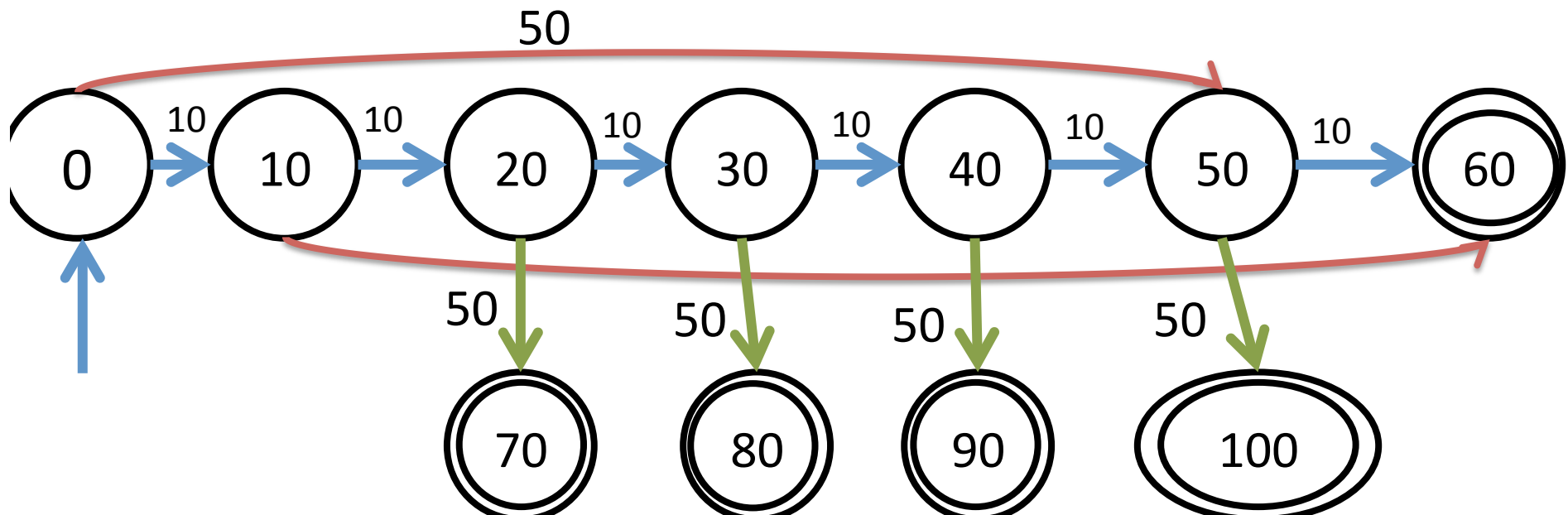
- 形式的定義の5要素

1.  $K$ : (状態集合)
2.  $\Sigma$ : アルファベット(入力として受け付ける記号)
3.  $q_0$ : 初期状態
4.  $\delta$ : 状態遷移
5.  $F$ : 終了状態

# 自動販売機の形式的定義

例) 自動販売機の場合

1.  $K = \{0, 10, 20, 30, 40, 50, 60, \dots, 100\}$
2.  $\Sigma = \{10, 50\}$
3.  $q_0 = 0$
4.  $\delta = \{f(0, 10)=10, f(0, 50)=50, f(10, 10)=20, f(10, 50)=60, \dots, f(50, 50)=100\}$
5.  $F = \{60, 70, 80, 90, 100\}$





### 3. 正規表現によるオートマトンの表現

- 正規表現を用いて記述することも出来る
  - 正規表現の例  
(+は「または」、\*は0回以上の連続を表す)

$(0+1)^*5$

この正規表現は「0もしくは1が0回以上連続し、最後が5で終わる数字列」を受理する

0005 : 受理

0105 : 受理

405 : 受理しない

# 本日の内容

- 講義：
  - 復習：決定性有限オートマトンの概要
  - 非決定性有限オートマトンの表現方法
  - grepコマンドで正規表現を扱う
- 演習：
  - 非決定性有限オートマトン
    - 受理する文字列を判定
    - 状態遷移図 $\leftrightarrow$ 正規表現の相互変換
  - grepコマンドで「吾輩は猫である」を解析

# 非決定性

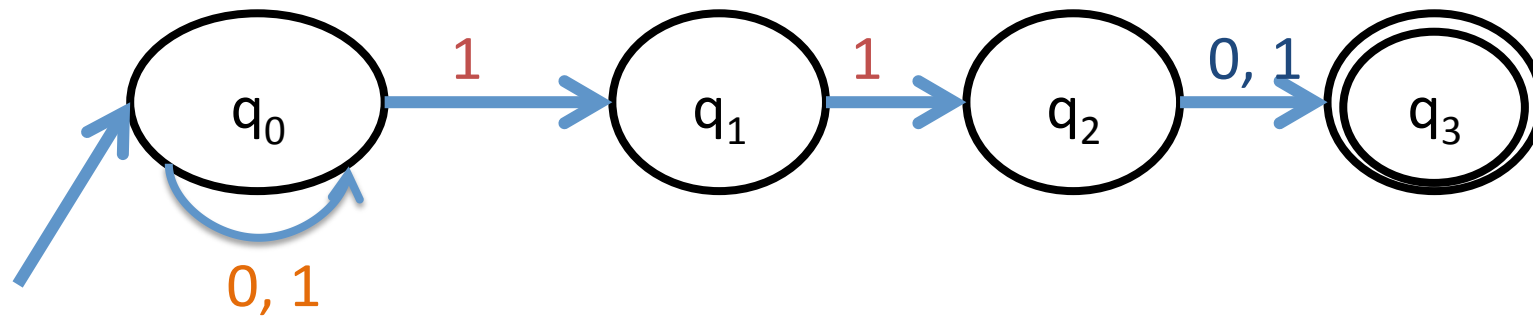
- 非決定性(NFA; Non-Deterministic Automaton)  
= 遷移先が一意に決まらないオートマトン
- 質問:  
以下の正規表現を状態遷移図で表現するには？

$(0+1)^*11(0+1)$

0もしくは1が0回以上連続して、その後1が2回連続し、0もしくは1で終わる

# NFAの例

- 正規表現  $(0+1)^*11(0+1)$  のオートマトン



状態  $q_0$  に 1 が入力された場合:

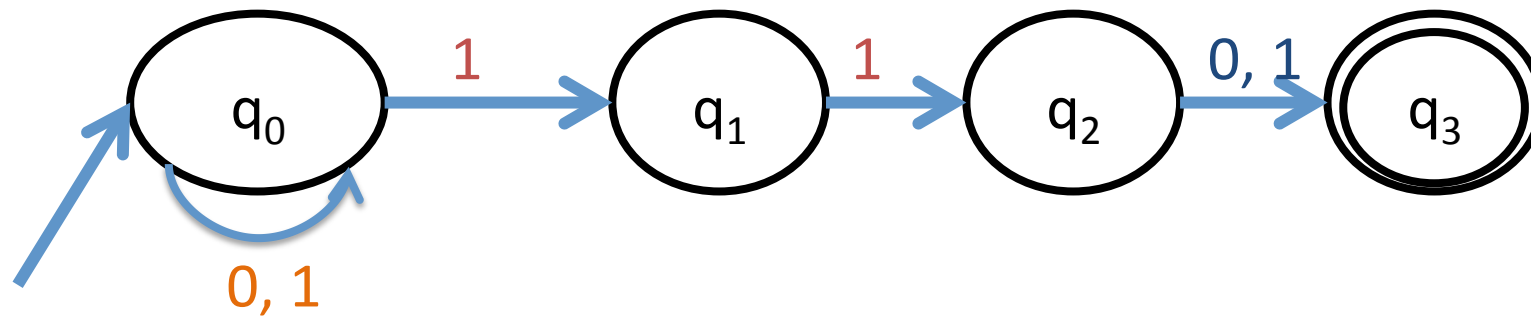
「 $q_0$  に遷移」するか「 $q_1$ 」に遷移するか一意に決まらない

# NFAの受理

- 受理
  - 入力に対して最終状態で止まる遷移方法が1つでもあれば受理
- 却下
  - どんな遷移方法を考えても最終状態にたどり着かないならば却下

# 受理する例/却下する例

- 例1: 0111(=受理)
  - $q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$ と進む遷移も考えられる
  - $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3$ と進む遷移が存在するので受理
- 例2: 0000(却下)
  - どの遷移を考えても終了状態 $q_3$ にたどり着けない



# 本日の内容

- 講義：
  - 復習：決定性有限オートマトンの概要
  - 非決定性有限オートマトンの表現方法
  - grepコマンドで正規表現を扱う
- 演習：
  - 非決定性有限オートマトン
    - 受理する文字列を判定
    - 状態遷移図 $\leftrightarrow$ 正規表現の相互変換
  - grepコマンドで「吾輩は猫である」を解析

# grepコマンドでの正規表現

- grepコマンド
  - テキスト中の条件に受理(マッチ)する行のみ抽出
  - 条件には正規表現を使用可
- grepコマンドのオプション
  - -Eオプションを付けて実行すると条件に正規表現を用いることが出来る
  - \$ grep -E '正規表現' ファイル名
  - -oオプションを付けると条件にヒットした部分のみ抽出可能(例はあとで)
  - \$ grep -E -o '正規表現' ファイル名



# 使える正規表現(一部抜粋)

- 文字列表現
  - . : 任意の1文字にヒット
  - ^ : 行の先頭
  - \$ : 行の末尾
- 繰り返し
  - \* : 直前の文字の0回以上の連続にヒット
  - + : 直前の文字の1回以上の連続にヒット
  - {x} : x回の連続にヒット
- 論理和演算子
  - [abcd] : abcdのいずれか1文字にヒット
  - 正規表現1 | 正規表現2 : 正規表現1, 2いずれかにヒット
  - [1-9] : 1から9までの数字にヒット
  - [a-z] : aからzまでの英小文字にヒット
  - [A-Z] : AからZまでの英大文字にヒット

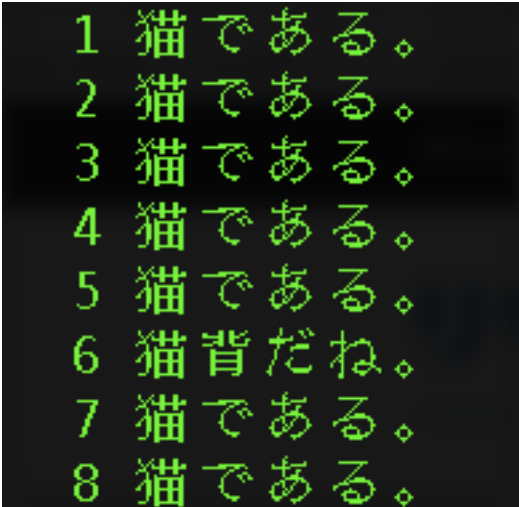
# コマンド例(1)

1. `grep -E '猫...。$' neko.txt`  
→「猫」から任意の文字列が4つ続き「。」で終わる文にヒット

```
1 吾輩は猫である。
2 彼は純粹の黒猫である。
3 しかし挨拶をしないと陰呑だと思ったから「吾輩は猫である。
4 車屋の黒はこの近辺で知らぬ者なき乱暴猫である。
5 元来黒は自慢をする丈にどこか足りないところがあって、彼の気
5 していればはなはだ御しやすい猫である。
6 「しかも曲っていらあ」「少し猫背だね。
7 吾輩は淡泊を愛する茶人的猫である。
8 吾輩は猫である。
```

## コマンド例(2)

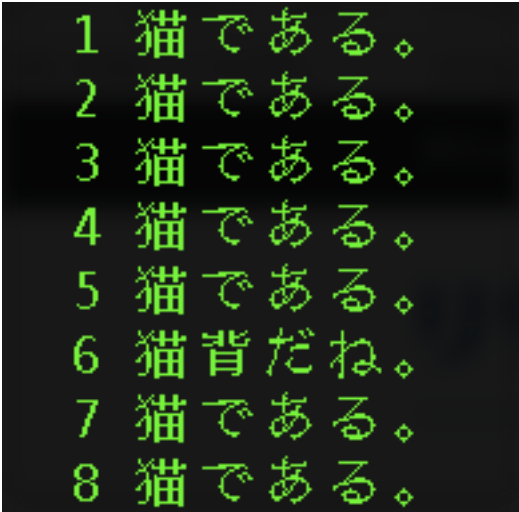
1. `grep -E -o '猫...。$' neko.txt`  
→「猫」から任意の文字列が4つ続き「。」で終わる部分のみ表示



```
1 猫である。  
2 猫である。  
3 猫である。  
4 猫である。  
5 猫である。  
6 猫背だね。  
7 猫である。  
8 猫である。
```

## コマンド例(2)

1. `grep -E -o '猫...。$' neko.txt`  
→「猫」から任意の文字列が4つ続き「。」で終わる部分のみ表示



```
1 猫である。  
2 猫である。  
3 猫である。  
4 猫である。  
5 猫である。  
6 猫背だね。  
7 猫である。  
8 猫である。
```

## コマンド例(3)

- `$ grep -E -o '猫|吾輩' neko.txt`  
→ 猫もしくは吾輩にヒット。ヒットした部分のみ表示

```
1 吾輩
2 猫
3 吾輩
4 猫
5 吾輩
6 吾輩
7 吾輩
8 吾輩
9 吾輩
```

# 出現回数のカウント(1)

- `grep -E -o -c '猫|吾輩' neko.txt`  
→ `-c`オプションでヒットする文字列数がカウントできる。
- `grep -E -o '猫|吾輩' neko.txt | wc -l`  
→ これでも同じ処理。

## 出現回数のカウント(2)

- `grep -E -o -c '猫|吾輩' neko.txt`  
→ `-c`オプションでヒットする文字列数がカウントできる。

633

- `grep -E -o '猫|吾輩' neko.txt | wc -l`  
→ これでも同じ処理。

## 出現回数のカウント(3)

- `grep -E -o '猫|吾輩' neko.txt | sort | uniq -c`  
→ 猫、吾輩の数をそれぞれカウントしたい場合にはsortしてからuniq -c コマンドにパイプする。

```
261 猫
481 吾輩
```

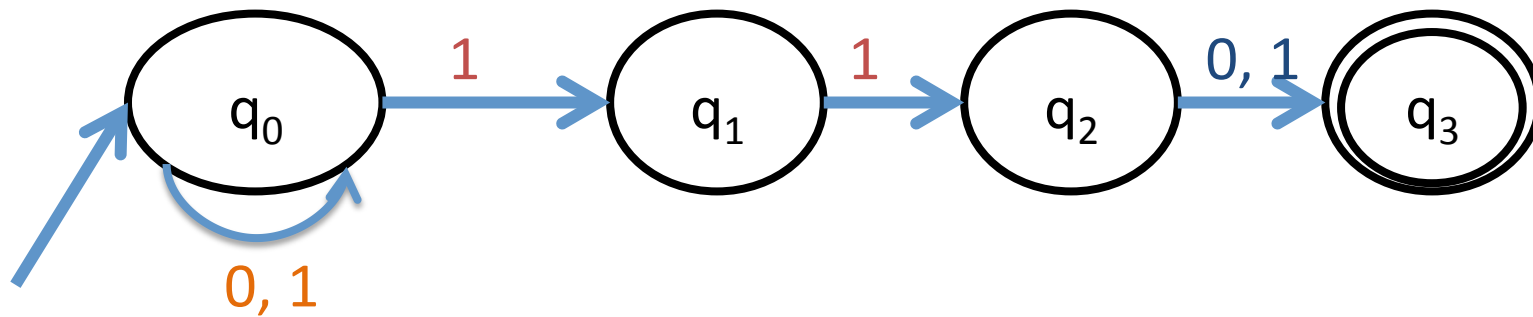


# 本日の内容

- 講義：
  - 復習：決定性有限オートマトンの概要
  - 非決定性有限オートマトンの表現方法
  - grepコマンドで正規表現を扱う
- 演習：
  - 非決定性有限オートマトン
    - 受理する文字列を判定
    - 状態遷移図 $\leftrightarrow$ 正規表現の相互変換
  - grepコマンドで「吾輩は猫である」を解析

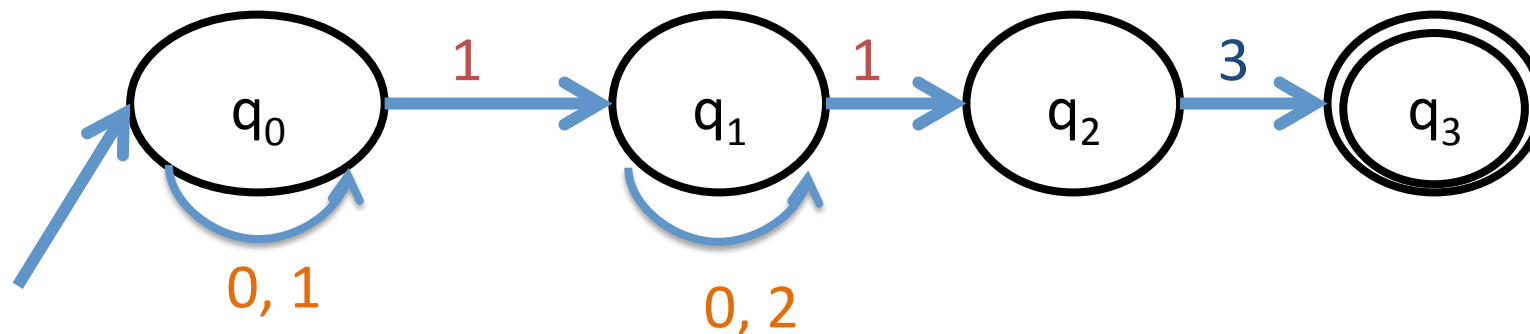
# 演習0(受理と却下)

- 以下の状態遷移図で表現される非決定性有限オートマトンが受理する文字列/却下する文字列(0と1の羅列)を5つずつ列挙しなさい。
- 「accept\_or\_not.txt」というファイル名で答えを保存し提出すること



# 演習1(状態遷移図と正規表現1)

- 以下の状態遷移図で表現される非決定性有限オートマトンを正規表現を用いて表現し、「regex.txt」というファイルに記述しなさい。  
(grepコマンドで用いる正規表現ではなく、+及び\*を用いた記法で記述すること)



## 演習2(状態遷移図と正規表現2)

- 以下の正規表現で記述されるオートマトンを、非決定性有限オートマトンの状態遷移図で記述せよ。状態名には $q_0, q_1, q_2$ を用いること。  
(提出不要)

$(a+b+c)^*dd^*a$

## 演習3(grepコマンド)

- 以下のコマンドで「吾輩は猫である」の全文を入手せよ  
`$ cp ~/ishigaki/neko.txt neko.txt`
- 講義資料内のコマンド例(1)からコマンド例(3)、出現回数のカウント(1)から(3)を順に入力し出力を確認せよ。  
(提出不要)

## 演習4(grep)

- grepコマンドを用いて吾輩は猫であるの全文のうち、「猫」と「主人」どちらの出現回数が多いか調べ、「neko\_shuzin.txt」というファイル名に使用したコマンドや出力結果をまとめよ。
- パイプを用いて1つのコマンドで記述すること

# 演習5

- 「吾輩は猫である」の文末4文字を正規表現を用いて取り出さない。sortコマンドやuniq -c等にパイプし、出現回数をカウントしない
- (以後optional課題)  
この結果を踏まえ、末尾4文字にはどのような特徴があるか考察しない。  
(ヒント:ジップ則、ロングテール等のキーワードで調べてみましょう)

# 早く終わった人

- 正規表現を使って「吾輩は猫である」にはどのような特徴があるか、末尾4文字以外以外に着目して分析してみましょう。

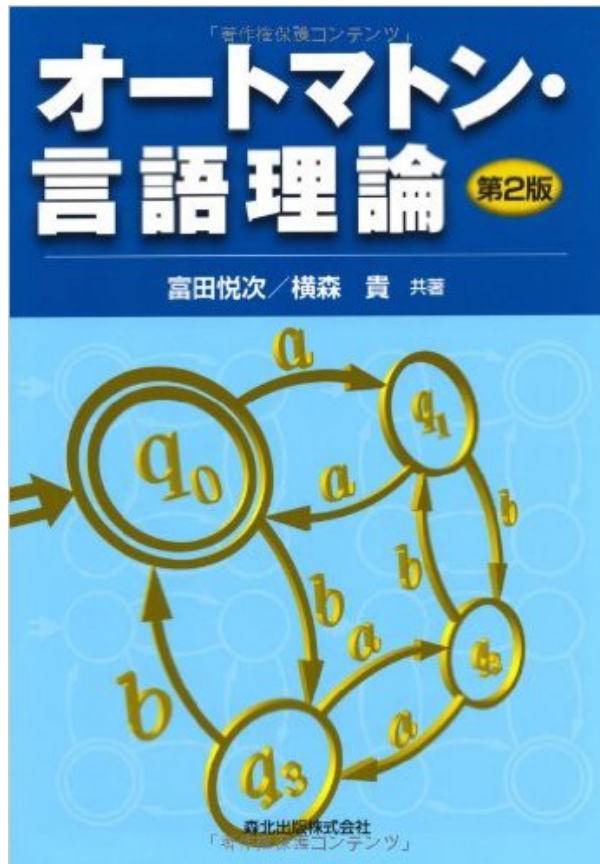
例) 末尾1文字が。ではなく」で終わっている文がある。「吾輩は猫である」では会話が全文の17%と多い。

```
$ grep -E -o '」$|。」$' neko.txt | sort | uniq -c  
7486 。  
1505 」
```



# さらに学びたい人

- 「オートマトン」「言語理論」の教科書が多数



オートマトン・言語理論  
富田悦次

# 次回

- コンパイラや計算機で‘言語’を処理するための基盤技術として使われる**文脈自由文法**について学びます
- 次々回：簡単なコンパイラを自作