有限オートマトンと正規表現

離散数学・オートマトン
2024 年後期
佐賀大学理工学部　只木進一

正規表現: Regular expressions

- 文字列の探索や置換で利用: Used for searching and replacing strings
- 柔軟にパターンを記述できる: Flexible pattern description
- 例: Example
  - "000" の繰り返しを含む: Contains repetitions of "000"
  - 数字が偶数個連続する: Contains an even number of digits
  - 指定した文字列の後ろに数字が付いているファイル名: File names with numbers following a specified string
- 多くのプログラミング言語やテキストエディタで利用できる: Available in many programming languages and text editors

例 1.1: ファイル名から日付部分を除く: Remove date from file name

```python
import re
fileList=[
    '1.1 Introduction_20201010.txt',
    '1.2 SetAndMappings_20211013.txt',
    '1.3 Relations_20100401.txt'
]
#日付以外の部分を取り出す
p: re.Pattern[str] = re.compile(r'(.*)_\d{8}(\.txt)')
for f in fileList:
    m: re.Match[str] | None = p.match(f)
    if m:
        filename: str = m.group(1)+m.group(2)
        print(filename)
```

```
1.1 Introduction.txt
1.2 SetAndMappings.txt
1.3 Relations.txt
```

例 1.2: ドメイン名を変更: Change domain name

```python
mail_address_list:list[str]=[
    'brown@example.com', 'page@hoge.com',
    't2013@example.edu', 'kate@hoge.com',
    's20235@foo.edu',    's19220@hoge.com'
]
pattern = r'(\S+)@(\S+)'
p: re.Pattern[str] = re.compile(pattern)
for mail in mail_address_list:
    m: re.Match[str] | None = p.match(mail)
    if m:
        domain = m.group(2)
        if domain == 'hoge.com':
            new_mail = m.group(1)+'@hogehoge.com'
            print(new_mail)

```

Python での正規表現: ごく一部
Regular expressions in Python: A very small part

- `.` **改行以外の任意の文字と一致**: Matches any character except a newline
- `\d` **数字と一致**: Matches any digit
- `\D` **数字以外と一致**: Matches any non-digit
- `\s` **スペースやタブ、改行などの空白文字と一致**: Matches any whitespace character
- `\S` **空白文字以外と一致**: Matches any non-whitespace character
- `*` **0 回以上の繰り返し**: 0 or more repetitions
- `+` **1 回以上の繰り返し**: 1 or more repetitions

正規表現の定義: 基礎
Defining Regular Expressions: Fundamentals

- $\mathrm{a} \in \Sigma$ に対して **a** は正規表現であり、その言語は $\{\mathrm{a}\}$ である:
  a is a regular expression for $\{\mathrm{a}\}$
  - 一文字からなる言語: a language consisting of a single character
- $\boldsymbol{\epsilon}$ は正規表現であり、その言語は $\{\epsilon\}$ である:
  $\boldsymbol{\epsilon}$ is a regular expression for $\{\epsilon\}$
  - 長さゼロの文字列からなる言語: a language consisting of a string of length zero
- $\boldsymbol{\emptyset}$ は正規表現であり、その言語は $\emptyset$ である:
  $\boldsymbol{\emptyset}$ is a regular expression for $\emptyset$
  - 要素を持たない言語: a language with no elements

正規表現の定義: 再帰
Defining Regular Expressions: Recursion

- α と β が、言語 L_α 及び L_β をそれぞれ表す正規表現のとき
  Assume α and β are regular expressions representing languages L_α and L_β respectively
  - $\alpha+\beta$ は $L_\alpha \cup L_\beta$ を表す正規表現: $\alpha+\beta$ is a regular expression representing $L_\alpha \cup L_\beta$
  - $\alpha\beta$ は $L_\alpha L_\beta$(連接:concatenation) を表す正規表現: $\alpha\beta$ is a regular expression representing $L_\alpha L_\beta$

$$L_\alpha L_\beta = \{uv | u \in L_\alpha, v \in L_\beta\} \tag{1.1}$$

  - α^* は Kleene 閉包: α^* is a Kleene closure : $L_\alpha^* = \bigcup_{k=0}^{\infty} L_\alpha^k$

$$L_\alpha^0 = \{\epsilon\}, L_\alpha^1 = L_\alpha, L_\alpha^{k+1} = L_\alpha L_\alpha^k \tag{1.2}$$

  - α^+ は正閉包:α^+ is a positive closure :$L_\alpha^+ = \bigcup_{k=1}^{\infty} L_\alpha^k$

例 1.3: $a, b \in \Sigma$

- $\mathbf{a}$ は言語 $\{a\}$ を表す: $\mathbf{a}$ represents the language $\{a\}$
- $\mathbf{b}$ は言語 $\{b\}$ を表す: $\mathbf{b}$ represents the language $\{b\}$
- $\mathbf{a} + \mathbf{b}$ は言語 $\{a, b\}$ を表す: $\mathbf{a} + \mathbf{b}$ represents the language $\{a, b\}$
- $\mathbf{ab}$ は言語 $\{ab\}$ を表す: $\mathbf{ab}$ represents the language $\{ab\}$
- $\mathbf{a}(\mathbf{a} + \mathbf{b})\mathbf{b}$ は言語 $\{aab, abb\}$ を表す: $\mathbf{a}(\mathbf{a} + \mathbf{b})\mathbf{b}$ represents the language $\{aab, abb\}$
- $(\mathbf{a} + \mathbf{b})^*$ は、a と b からなる、長さゼロ以上の文字列全体からなる言語を表す: $(\mathbf{a} + \mathbf{b})^*$ represents the language consisting of all strings of zero or more characters from a and b

例 1.4:

$$(\mathbf{0}+\mathbf{1})\mathbf{1}(\mathbf{0}+\mathbf{1})$$

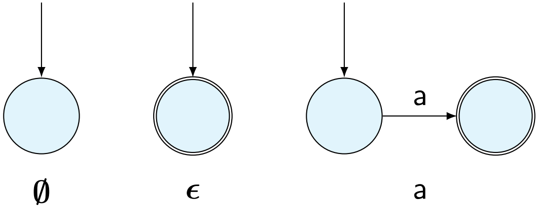
$$\mathbf{0}(\mathbf{0}+\mathbf{1})^*\mathbf{0}$$

正規表現と有限オートマトン Regular Expressions and Finite State Automata

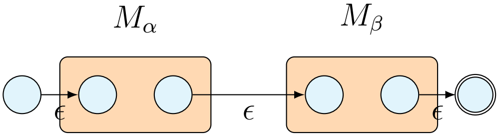
- 任意の正規表現を受理言語とする有限オートマトンを構成することができる: A finite automaton can be constructed for accepting the language of any regular expression
- 任意の有限オートマトンの受理言語を表す正規表現を構成することができる: A regular expression can be constructed for representing the accepting language of any finite automaton
- つまり、有限オートマトンの受理言語は正規表現で表される: In other words, the accepting language of a finite automaton is represented by a regular expression

正規表現から FA へ: 基礎的表現
From Regex to FA: Basic Representations

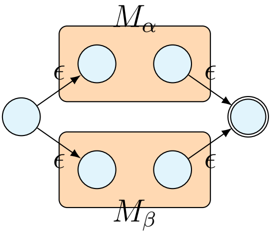
- **正規表現の構成を順を追って FA で表現**: Constructing regular expressions in order to represent them in finite automata
- **基礎**:Fundamentals: $a, b \in \Sigma$

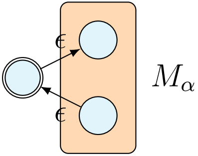


和、連接、Kleene 閉包
Union, Concatenation, Kleene Closure



$\boldsymbol{\alpha\beta}$

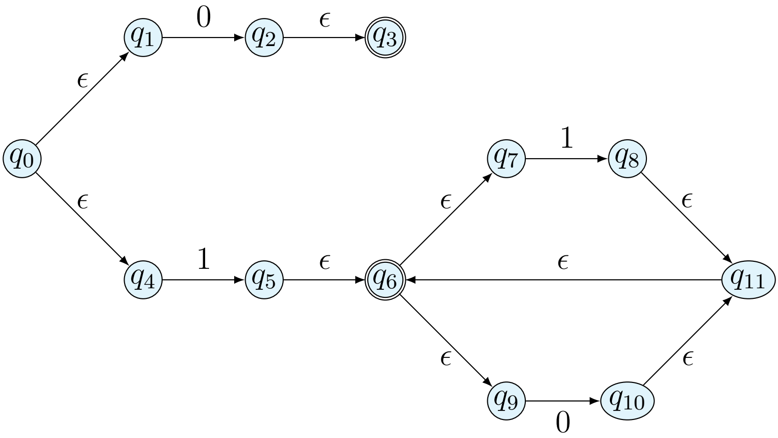




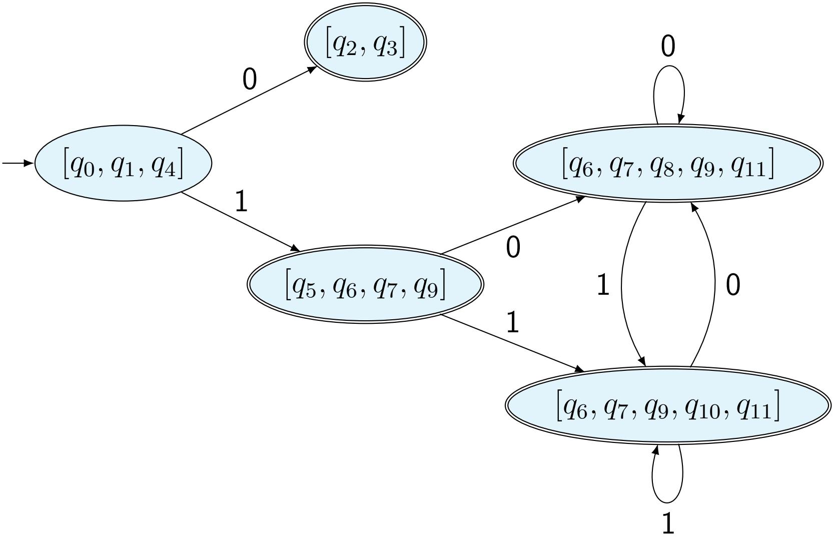
$\boldsymbol{\alpha}^*$

$\boldsymbol{\alpha} + \boldsymbol{\beta}$

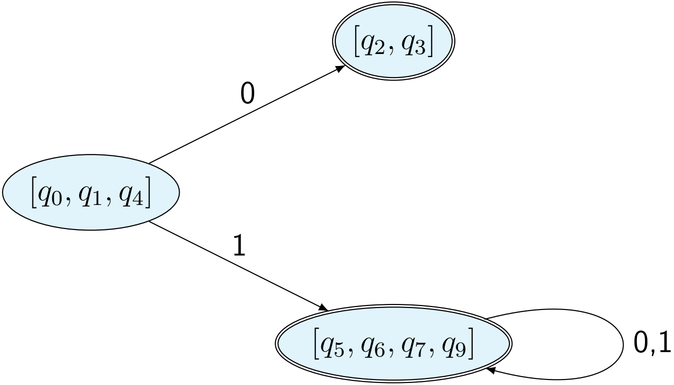
例 3.1: $\mathbf{0} + \mathbf{1}\,(\mathbf{0} + \mathbf{1})^*$



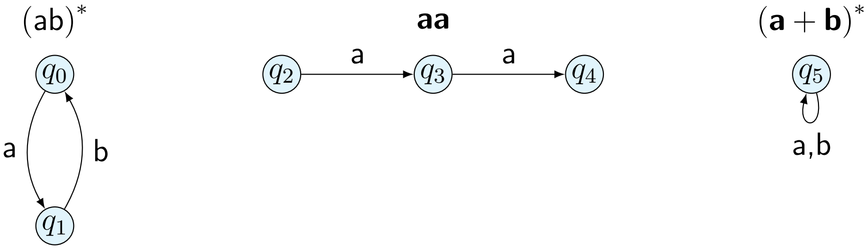
DFA へ変換: NFA to DFA

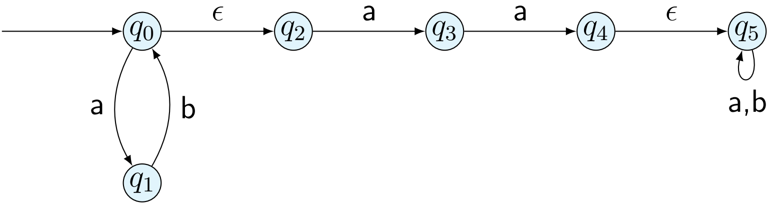


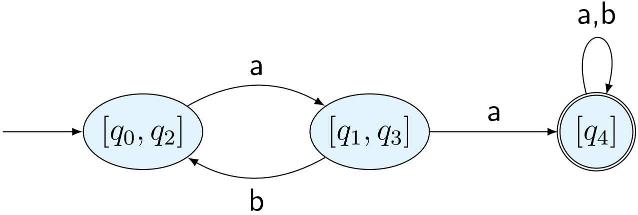
DFA を最小化: Minimizing DFA



例 3.2: $(\mathbf{ab})^* \, \mathbf{aa} \, (\mathbf{a} + \mathbf{b})^*$

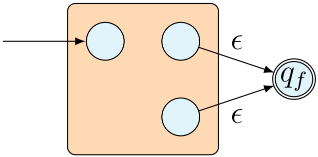



q_0
q_1
q_2
q_3
q_4
q_5
ϵ
a
a
ϵ
a
b
a,b

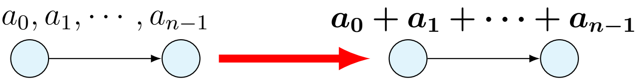

$[q_0, q_2]$
$[q_1, q_3]$
$[q_4]$
a
b
a
a,b

有限オートマトンの受理言語を正規表現で構成: From FA to Regex

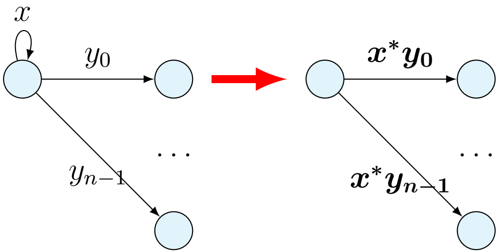
- Step1: 新たに一つの終状態 q_f を追加し、それのみが終状態とする: Add a new final state q_f and make it the only final state



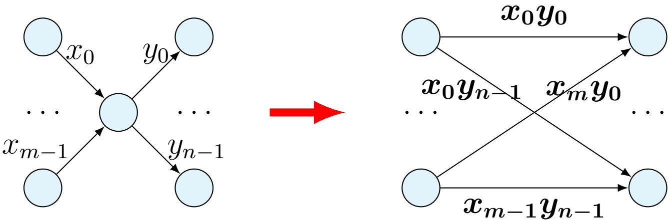
- step2: rule1,2,3 の順に適用する: Apply rule1,2,3 in order
- rule1: 同じ状態遷移を引き起こす入力に対して、正規表現の和を対応つける: Correspond a regular expression union for inputs causing the same state transition



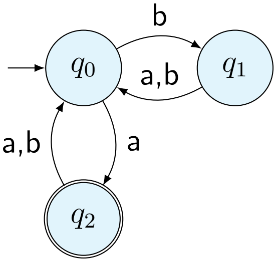
- rule2: **ループの遷移を次の遷移の前に連接する**: Concatenate the transition of the loop before the next transition

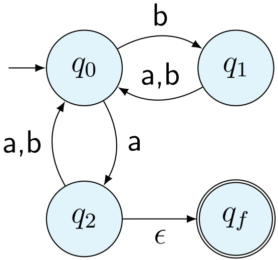


- rule3: **連続する遷移を、途中の状態を削除して連接とする**: Concatenate the consecutive transitions by removing intermediate states



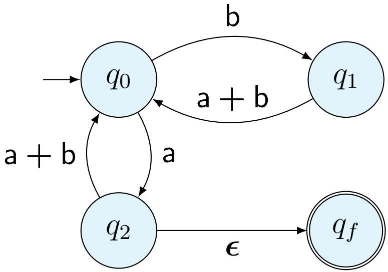
例 4.1:





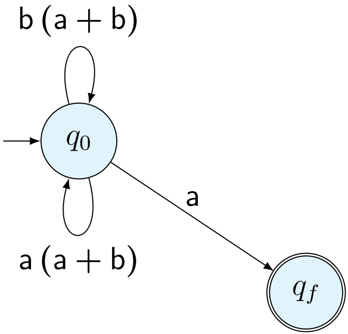
rule 1

同じ遷移を起こす入力を正規表現の和に変換: Strings that cause the same transition are combined with a plus sign.



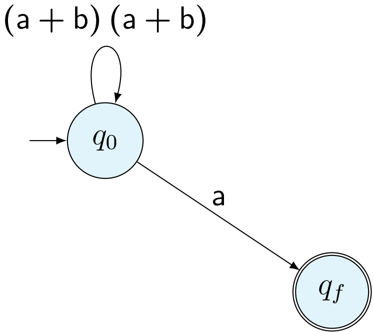
rule 3

- rule 2 に相当するループが: Loops that are equivalent to rule 2
  - q_0 から q_1 を経て、q_0 へ戻る経路をループに: Loop from q_0 to q_1 and back to q_0
  - q_0 から q_2 を経て、q_0 へ戻る経路をループにするとともに、a_f への遷移を残す: Loop from q_0 to q_2 and back to q_0 and keep the transition to q_f



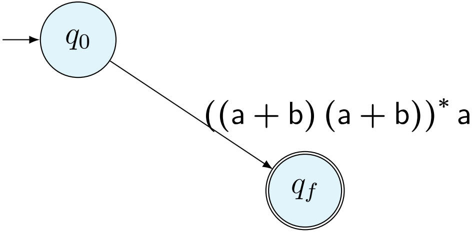
rule 1: ２回目

- q_0 の２つのループの表現の和を作る: Combine the representation of the two loops of q_0

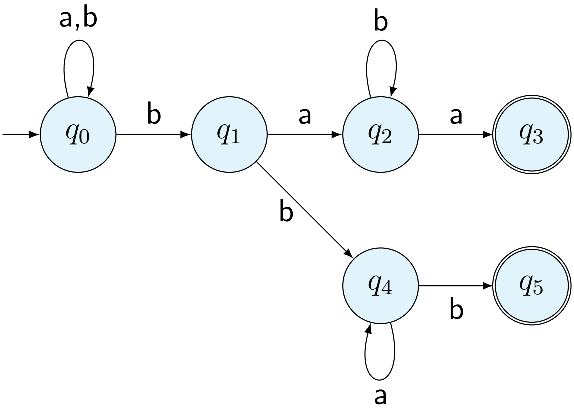


rule 2

- q_0 の２つのループの表現の和を作る: Combine the representation of the two loops of q_0

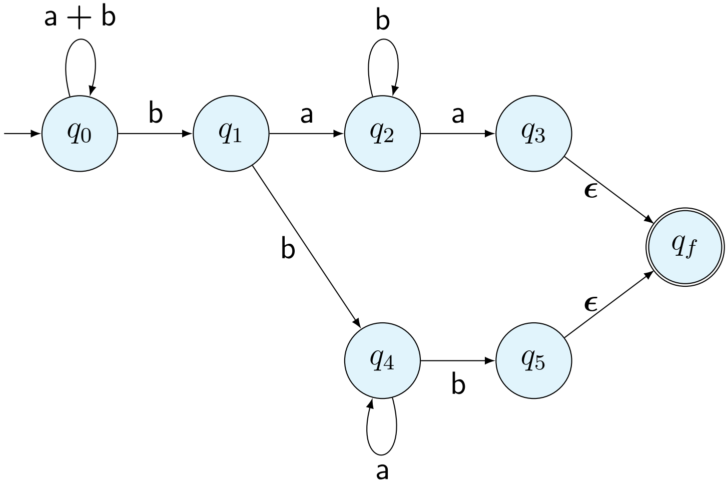


例 4.2:

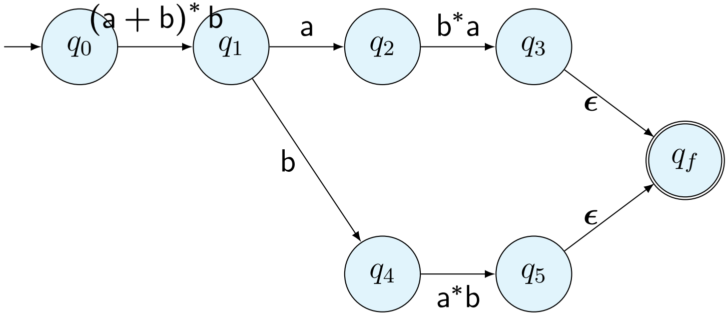


rule 1

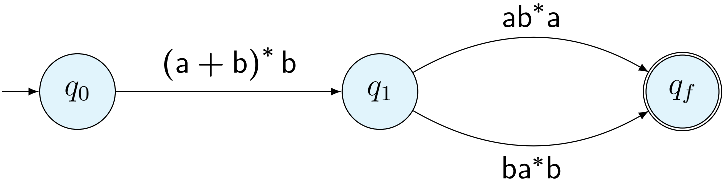
q_f を追加し、rule1 を適用



rule 2



rule 3



$$(a+b)^*b\,(ab^*a+ba^*b)$$

正規表現に対する反復補題: Pumping Lemma

- L を正規言語とする: Let L be a regular language
- L に対して、ある定数 n が存在し、$|w| \geq n$ となる任意の $w \in L$ に対し、以下のように $w = xyz$ と分解できる。: For any $w \in L$ with $|w| \geq n$, there exists a decomposition $w = xyz$ such that
  - $y \neq \epsilon$
  - $|xy| \leq n$
  - $\forall k \geq 0,\ xy^k z \in L$

反復補題の証明: Proof of Pumping Lemma

- L を正規言語とし、対応する DFA を M とする。: Let L be a regular language and M be the corresponding DFA
  - $L = L(M)$
  - M の状態数を n とする: Let n be the number of states in M

$$w = a_1 a_2 \cdots a_m \in L,\ (m \geq n,\ a_i \in \Sigma)$$
$$p_i = \delta(q_0, a_1 a_2 \cdots a_i),\ (p_0 = q_0)$$

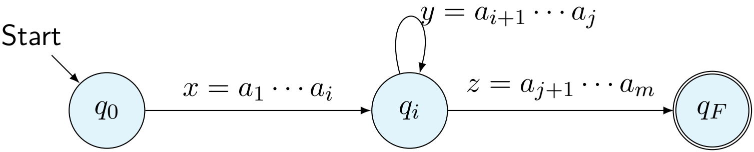
- $p_0, p_1, \cdots, p_n$ には少なくとも一つの状態が 2 回以上現れる: At least one state appears twice in $p_0, p_1, \cdots, p_n$

$$p_i = p_j,\ (0 \leq i < j \leq n)$$

- $w = xyz$ と分解する: Decompose $w = xyz$

$$\begin{array}{ccc} x = a_1 a_2 \cdots a_i, & y = a_{i+1} a_{i+2} \cdots a_j, & z = a_{j+1} a_{j+2} \cdots a_m \\ |x| = i \geq 0, & |y| = j - i > 0, & |z| = m - j \geq 0 \end{array}$$

- $xy^k z \in L \ (0 \leq k)$



反復補題の応用: Applications of Pumping Lemma

$L_{01} = \{0^m 1^m | m \geq 0\}$ **は正規言語でない**: $L_{01} = \{0^m 1^m | m \geq 0\}$ is not a regular language

- L_{01} **が正規言語であると仮定**: Assume L_{01} is a regular language
- **反復補題に現れる** n **に対して、**$w = 0^n 1^n$ **を考える**: Consider $w = 0^n 1^n$ for the n in the pumping lemma
- $w = xyz$、$|xy| \leq n$ **より、**$y = 0^k$ $(1 \leq k \leq n)$

$$xy^0 z = xz = 0^{n-k} 1^n \notin L_{01}$$

- L_{01} **は正規言語でない**: L_{01} is not a regular language