コンパイラ

湯淺太一

第２章

字　句　解　析

文字列の計算

アルファベット：言語を構成する文字の（有限）集合

文字列: 文字の並び

空の文字列: 文字を一つも含まない文字列ε

文字列の連結$s \cdot t$

例：$\mathtt{aa} \cdot \mathtt{bc} = \mathtt{aabc}$

結合則$s \cdot (t \cdot u) = (s \cdot t) \cdot u$が成り立つ

$\rightarrow$ $s \cdot t \cdot u$と書いてよい

$\rightarrow$ stuと略記することもある

文字列のべき乗

$s^0 = \varepsilon$

$s^n = s \cdot s^{n-1} \quad (n = 1, 2, 3, \ldots)$

公式

$s^1 = s \cdot s^0 = s \cdot \varepsilon = s$

$s^{i+j} = s^i \cdot s^j$

文字列集合の連結

$A \cdot B = \{s \cdot t \,|\, s \in A,\ t \in B\}$

例：$\{\mathsf{a}, \mathsf{bc}\} \cdot \{\mathsf{a}, \mathsf{c}\} = \{\mathsf{aa}, \mathsf{ac}, \mathsf{bca}, \mathsf{bcc}\}$

結合則$A \cdot (B \cdot C) = (A \cdot B) \cdot C$が成り立つ

$\rightarrow A \cdot B \cdot C$と書いてよい

$\rightarrow ABC$と略記することがある

文字列集合のべき乗

$A^0 = \{\varepsilon\}$

$A^n = A \cdot A^{n-1} = \underbrace{A \cdot A \cdot \cdots \cdot A}_{n} \quad (n = 1, 2, 3, \ldots)$

例：$A = \{\mathsf{a}, \mathsf{bc}\}$のとき

$A^3 = A \cdot A \cdot A = \{\mathsf{aaa}, \mathsf{aabc}, \mathsf{abca}, \mathsf{abcbc}, \mathsf{bcaa}, \mathsf{bcabc}, \mathsf{bcbca}, \mathsf{bcbcbc}\}$

公式

$A^1 = A \cdot A^0 = A \cdot \{\varepsilon\} = A$

$A^{i+j} = A^i \cdot A^j = \underbrace{\overbrace{A \cdot A \cdot \cdots \cdot A}^{i} \cdot \overbrace{A \cdot A \cdot \cdots \cdot A}^{j}}_{i+j}$

$\{\varepsilon\}$と空集合

$\{\varepsilon\} \cdot A = A \cdot \{\varepsilon\} = A$

$\phi \cdot A = A \cdot \phi = \phi$

文字列集合の正の閉包(positive closure)

$$A^+ = A^1 \cup A^2 \cup A^3 \cup \cdots = \bigcup_{i=1}^{\infty} A^i$$

例：

$$\{\mathtt{a}\}^+ = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \ldots\}$$

$$\{\mathtt{a}, \mathtt{b}\}^+ = \{\mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{aaa}, \ldots\}$$

文字列集合の閉包(closure)

$$A^* = \{\varepsilon\} \cup A^+ = A^0 \cup A^1 \cup A^2 \cup A^3 \cup \cdots = \bigcup_{i=0}^{\infty} A^i$$

例：

$$\{\mathtt{a}\}^* = \{\varepsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \ldots\}$$

公式

$$A^* \cdot A = A \cdot A^* = A^+$$

$$A^* \cdot A^+ = A^+ \cdot A^* = A^+$$

$$A^* \cdot A^* = A^*$$

例：C 言語における 10 進定数

$$\{\mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}\} \cdot \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}\}^*$$

正規表現

アルファベット $\Sigma = \{a_1, \cdots, a_n\}$ 上の正規表現

1. ε
2. $a_i \quad (a_i \in \Sigma)$
3. $r_1 \,|\, r_2$ （r_1, r_2 は正規表現）
4. $r_1 \cdot r_2$ （r_1, r_2 は正規表現）
5. r^* （r は正規表現）

$L(r)$：正規表現 r が表す文字列集合

$L(\varepsilon) = \{\varepsilon\}$

$L(a_i) = \{a_i\}$

$L(r_1 \,|\, r_2) = L(r_1) \cup L(r_2)$

$L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$

$L(r^*) = L(r)^*$

例：$\mathtt{a} \cdot ((\mathtt{a} \,|\, \mathtt{b})^*)$ は，$\{\mathtt{a}, \mathtt{b}\}$ 上の正規表現．　$\mathtt{a}\,(\mathtt{a} \,|\, \mathtt{b})^*$ と略記

$$
\begin{aligned}
&L(\mathtt{a}(\mathtt{a} \,|\, \mathtt{b})^*) \\
&\quad = L(\mathtt{a}) \cdot L((\mathtt{a} \,|\, \mathtt{b})^*) = L(\mathtt{a}) \cdot (L(\mathtt{a} \,|\, \mathtt{b}))^* = L(\mathtt{a}) \cdot (L(\mathtt{a}) \cup L(\mathtt{b}))^* = \{\mathtt{a}\} \cdot (\{\mathtt{a}\} \cup \{\mathtt{b}\})^* \\
&\quad = \{\mathtt{a}\} \cdot \{\mathtt{a}, \mathtt{b}\}^* = \{\mathtt{a}\} \cdot \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{aaa}, \ldots\} \\
&\quad = \{\mathtt{a}, \mathtt{aa}, \mathtt{ab}, \mathtt{aaa}, \mathtt{aab}, \mathtt{aba}, \mathtt{abb}, \mathtt{aaaa}, \ldots\}
\end{aligned}
$$

正規表現rの解釈

$\varepsilon \mid r$：rが省略可能

a_i：a_i自体

$r_1 \mid r_2$：r_1またはr_2

$r_1 \cdot r_2$：r_1のあとにr_2

r^*：rが0個以上

略記法

$r_{opt} = \varepsilon \mid r$

$r^+ = r \cdot r^*$

C言語の字句要素

識別子: $\alpha\,(\alpha \mid \delta)^*$

キーワード: `auto` | `break` | `case` | `char` | $\cdots$

演算子: `+` | `&` | `=` | `+=` | $\cdots$

区切り記号: `,` | `;` | `(` | `)` | $\cdots$

10進定数: (`1` | $\cdots$ | `9`) $\delta^*\langle$オプション$\rangle$

浮動小数点定数: $((\delta^+ \,.\, \delta^* \mid .\, \delta^+)\langle$指数部$\rangle_{opt} \mid \delta^+\langle$指数部$\rangle)$ (`F` | `f` | `L` | `l`)$_{opt}$

文字定数: `L`$_{opt}$ ’ (\$\langle$文字$\rangle$ | $\langle$\と’と改行以外$\rangle$)$^+$ ’

ここで,

$\alpha =$ `_` | `a` | `b` | $\cdots$ | `z` | `A` | `B` | $\cdots$ | `Z`

$\delta =$ `0` | `1` | `2` | $\cdots$ | `9`

$\langle$オプション$\rangle = ((\mathtt{U} \mid \mathtt{u})\,(\mathtt{L} \mid \mathtt{l})_{opt} \mid (\mathtt{L} \mid \mathtt{l})\,(\mathtt{U} \mid \mathtt{u})_{opt})_{opt}$

$\langle$指数部$\rangle = (\mathtt{E} \mid \mathtt{e})\ \mathtt{-}_{opt}\ \delta^+$

有限オートマトン (finite automaton)

有限個の状態を持ち，文字を 1 文字ずつ読み込むことによって，自分自身の状態を変化させる，一種の仮想計算機

FA は五つ組 $\langle \Sigma, Q, \Delta, s, F \rangle$ で定義

Σ: アルファベット

Q: 状態全体の集合

Δ: 状態遷移 $\langle q_1, a, q_2 \rangle$ の集合

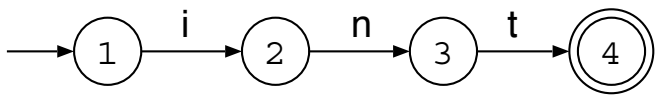
状態 q_1 で a を読んだら状態 q_2 に遷移

s: 初期状態 $(s \in Q)$

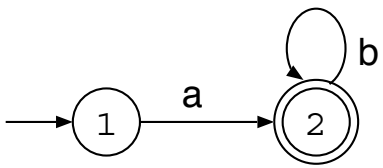
F: 受理状態の集合 $(F \subset Q)$

状態遷移図

$\mathrm{M1} = \langle \Sigma, \{1, 2, 3, 4\}, \{\langle 1, \mathtt{i}, 2 \rangle, \langle 2, \mathtt{n}, 3 \rangle, \langle 3, \mathtt{t}, 4 \rangle\}, 1, \{4\} \rangle$



$\mathrm{M2} = \langle \Sigma, \{1, 2\}, \{\langle 1, \mathtt{a}, 2 \rangle, \langle 2, \mathtt{b}, 2 \rangle\}, 1, \{2\} \rangle$

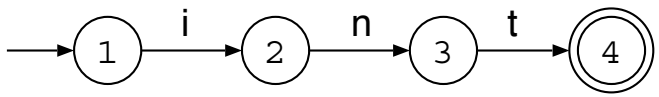


$M = \langle \Sigma, Q, \Delta, s, F \rangle$: FA
$\alpha = a_1 a_2 \cdots a_n$: 入力文字列
次の条件を満たす状態列$q_0, q_1, \ldots, q_n$が存在するとき，Mはαを受理(accept)する．

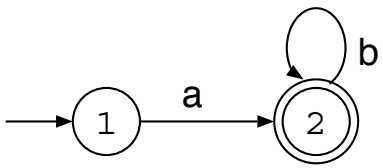
$$s = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n \in F$$

$L(M)$: Mが受理する文字列全体
　$\rightarrow$ Mが受理する言語(language)

例：
$L(\mathrm{M1}) = \{\texttt{int}\}$



$L(\mathrm{M2}) = \{\mathsf{a}, \mathsf{ab}, \mathsf{abb}, \mathsf{abbb}, \ldots\} = L(\mathsf{ab}^*)$



$L(M) = L(M')$のとき，MとM'は等価(equivalent)

決定性(deterministic)有限オートマトン

任意の$q \in Q$と$a \in \Sigma$に対して，$\langle q, a, q' \rangle$の形の状態遷移が高々一つ

DFA $\langle \Sigma, Q, \Delta, s, F \rangle$の状態遷移関数

$$\delta(q, a) = \begin{cases} q' & (\langle q, a, q' \rangle \in \Delta \text{のとき}) \\ \perp & (\text{その他の場合}) \end{cases}$$

δが与えられれば，

$$\Delta = \{\langle q, a, \delta(q, a) \rangle \mid q \in Q, a \in \Sigma, \delta(q, a) \neq \perp\}$$

DFAを$\langle \Sigma, Q, \delta, s, F \rangle$と表記することがある．

適用例(1)：受理するかどうかの判定

入力文字列: $a_1 a_2 \cdots a_n$

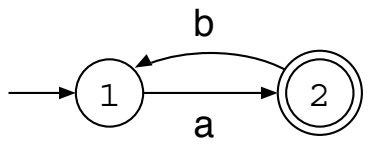
1. $i \leftarrow 1$，$q \leftarrow s$
2. $i \leq n$かつ$\delta(q, a_i) \neq \perp$ならば，
   $q \leftarrow \delta(q, a_i)$，$i \leftarrow i + 1$
   とし，このステップ2を繰り返す．
3. $i > n$かつ$q \in F$であればyesと表示し，そうでなければnoと表示する．

適用例(2)：文字列の切り出し

入力文字列: $a_1 a_2 \cdots a_n$

1. $k \leftarrow 0$，$i \leftarrow 0$，$q \leftarrow s$（初期状態）
2. $q \in F$であれば，$k \leftarrow i$
3. $i \leftarrow i + 1$
4. $\delta(q, a_i) \neq \perp$なら，$q \leftarrow \delta(q, a_i)$とし，ステップ2へ戻る．そうでなければ，$k$の値を返す．

例：

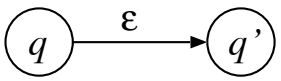


入力文字列がababbのとき

$$1 \xrightarrow{\texttt{a}} 2_{(k=1)} \xrightarrow{\texttt{b}} 1 \xrightarrow{\texttt{a}} 2_{(k=3)} \xrightarrow{\texttt{b}} 1 \xrightarrow{\texttt{b}} \perp$$

非決定性有限オートマトン

ε遷移　$\langle q, \varepsilon, q' \rangle$



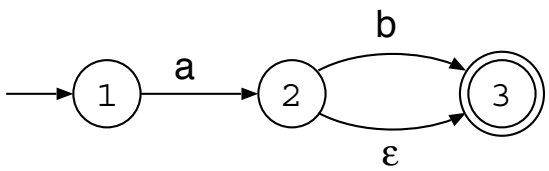
状態qに達したときに，入力文字列と無関係に，
q'へ遷移してよい．qに留まってもよい．

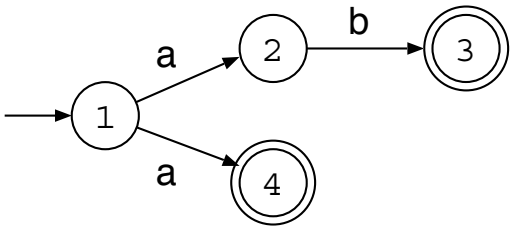
非決定性(nondeterministic)有限オートマトン(NFA)
次のいずれかを満たすFA $\langle \Sigma, Q, \Delta, s, F \rangle$

1. Δがε遷移を含む．
2. ある$q \in Q$と$a \in \Sigma$に対して，$\langle q, a, q' \rangle$の形の状態遷移が，Δの中に複数存在する．

例

$\mathrm{M4} = \langle \Sigma, \{1, 2, 3\}, \{\langle 1, \mathtt{a}, 2 \rangle, \langle 2, \mathtt{b}, 3 \rangle, \langle 2, \varepsilon, 3 \rangle\}, 1, \{3\} \rangle$
$\mathrm{M5} = \langle \Sigma, \{1, 2, 3, 4\}, \{\langle 1, \mathtt{a}, 2 \rangle, \langle 2, \mathtt{b}, 3 \rangle, \langle 1, \mathtt{a}, 4 \rangle\}, 1, \{3, 4\} \rangle$





次の条件をすべて満たす経路

$$q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} \cdots \xrightarrow{x_m} q_m$$

が存在するとき，NFA $\langle \Sigma, Q, \Delta, s, F \rangle$ は，文字列 $\alpha = a_1 a_2 \cdots a_n$ を受理する．

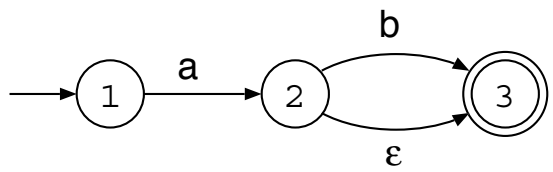
1. $q_0 = s$
2. $\langle q_{i-1}, x_i, q_i \rangle \in \Delta \ (i = 1, 2, \ldots, m)$
3. $x_1, x_2, \ldots, x_m$ から ε を除去すると α と一致する．
4. $q_m \in F$

例:

$L(\mathtt{M4}) = \{\mathtt{a}, \mathtt{ab}\}$

$\mathtt{a}$ に対して: $1 \xrightarrow{\mathtt{a}} 2 \xrightarrow{\varepsilon} 3$

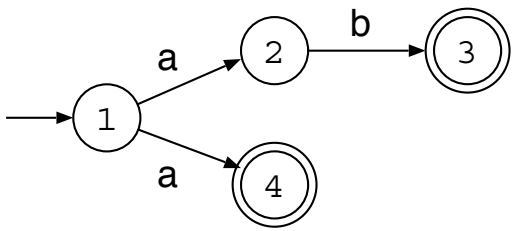
$\mathtt{ab}$ に対して: $1 \xrightarrow{\mathtt{a}} 2 \xrightarrow{\mathtt{b}} 3$



$L(\mathtt{M5}) = \{\mathtt{a}, \mathtt{ab}\}$

$\mathtt{a}$ に対して: $1 \xrightarrow{\mathtt{a}} 4$

$\mathtt{ab}$ に対して: $1 \xrightarrow{\mathtt{a}} 2 \xrightarrow{\mathtt{b}} 3$



正規表現からDFAへの変換

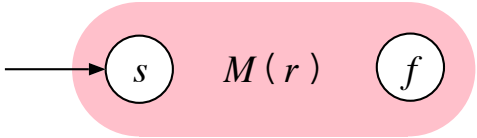
r：正規表現
M：FA
$L(r) = L(M)$のとき，rとMは等価

正規表現rを，等価なDFAに変換

1. rを，等価なNFAに変換
2. NFAを，等価なDFAに変換
3. DFAを，状態数最小の等価なDFAに変換

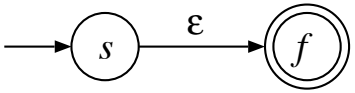
正規表現からNFAへの変換

$M(r)$: 正規表現rに対する標準形NFA

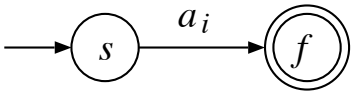


1. 初期状態への遷移は存在しない
2. 受理状態は一つだけ
3. 受理状態からの遷移は存在しない

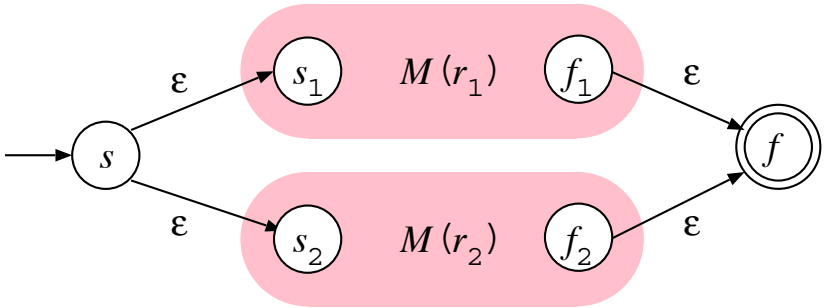
(1) $M(\varepsilon)$



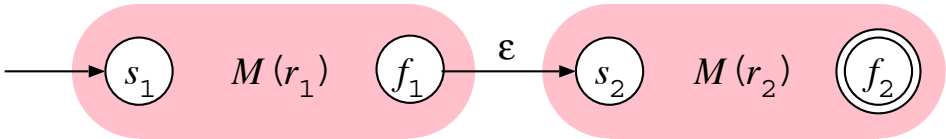
(2) $M(a_i)$



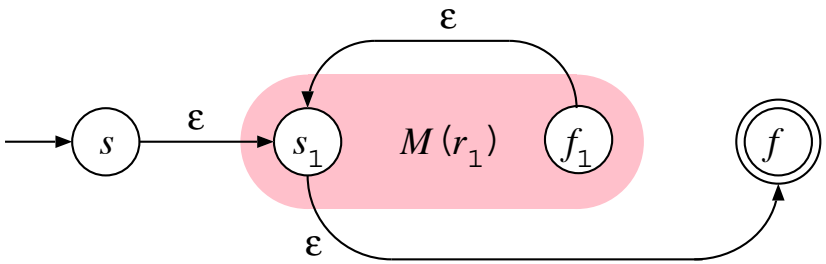
(3) $M(r_1 \mid r_2)$



(4) $M(r_1 \cdot r_2)$



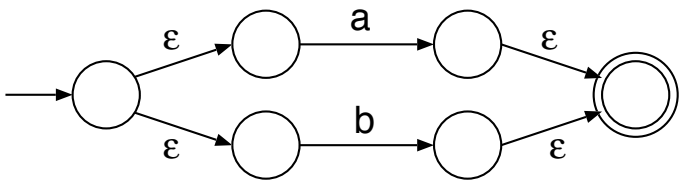
(5) $M(r_1^*)$



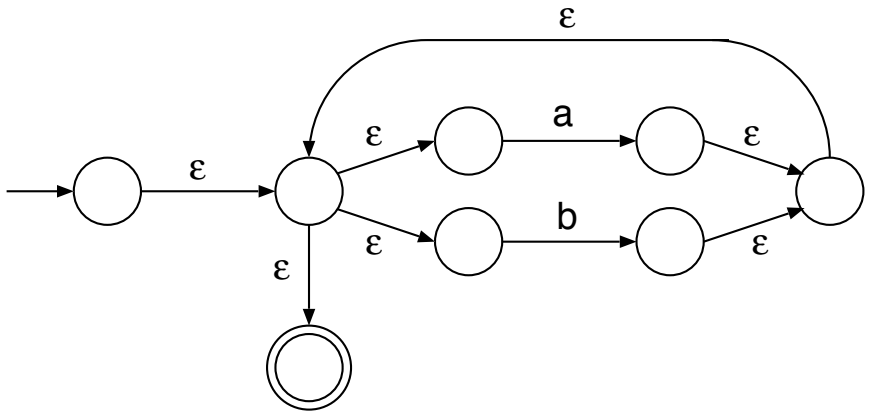
例：$M((\mathtt{a}\,|\,\mathtt{b})^*\mathtt{ab})$

(1) $M(\mathtt{a})$ と $M(\mathtt{b})$

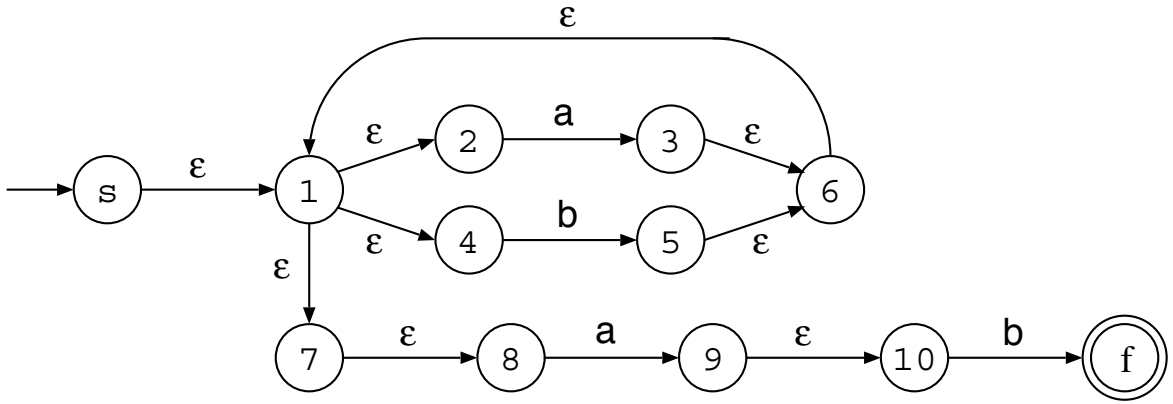
(2) $M(\mathtt{a}\,|\,\mathtt{b})$



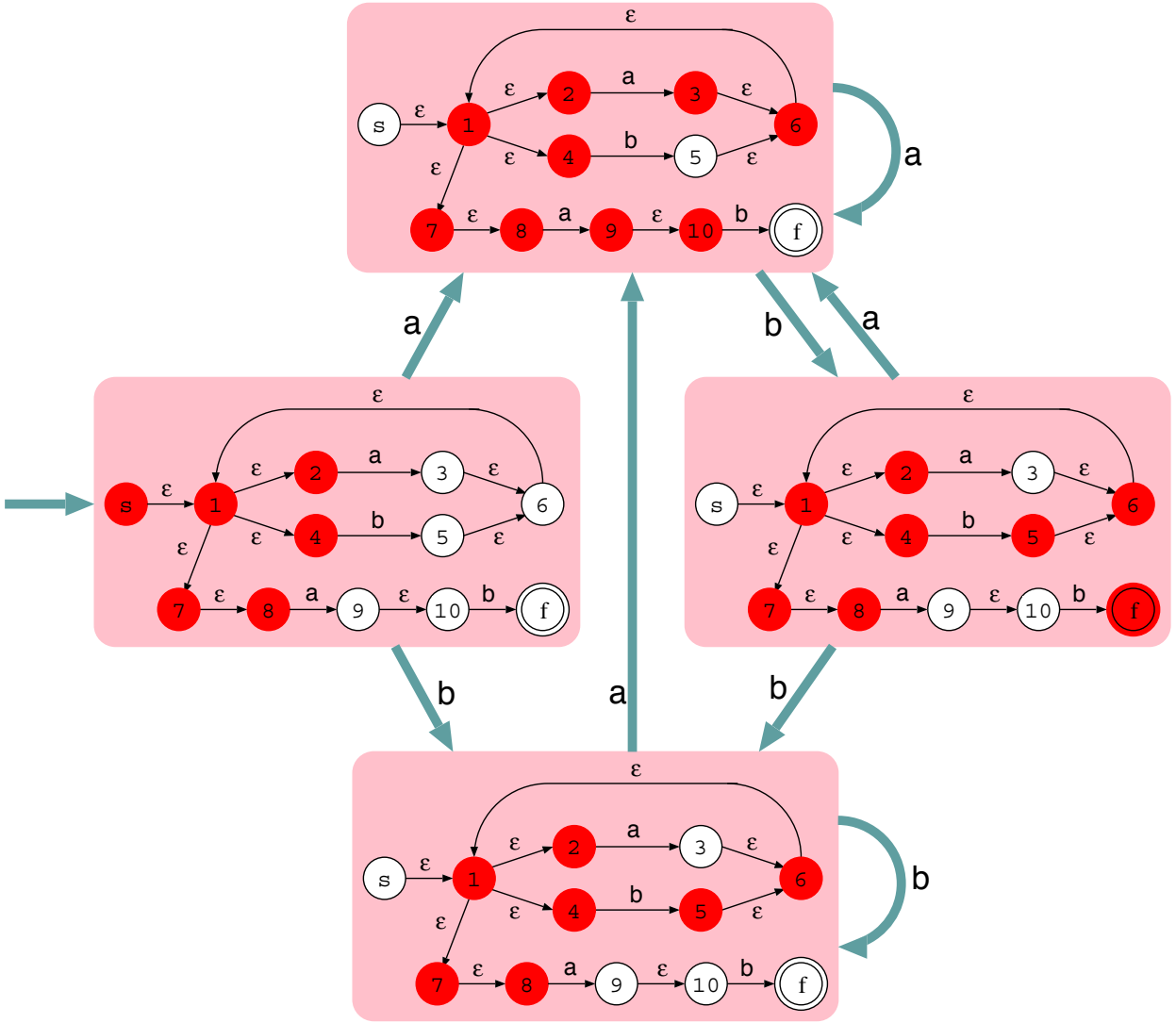
(3) $M((\mathtt{a}\,|\,\mathtt{b})^*)$

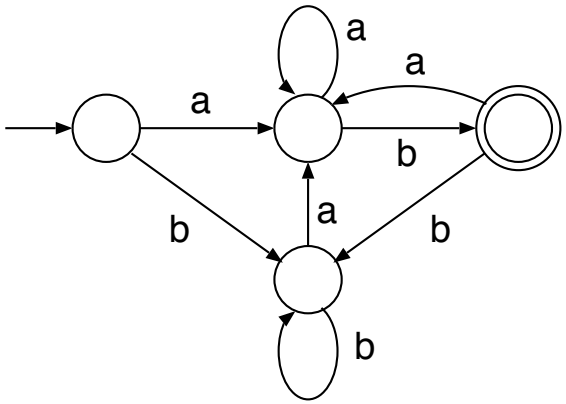


(4) $M((\mathtt{a}\,|\,\mathtt{b})^*\mathtt{ab})$

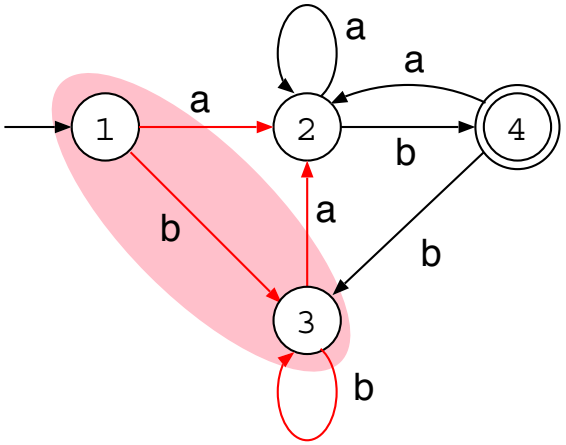


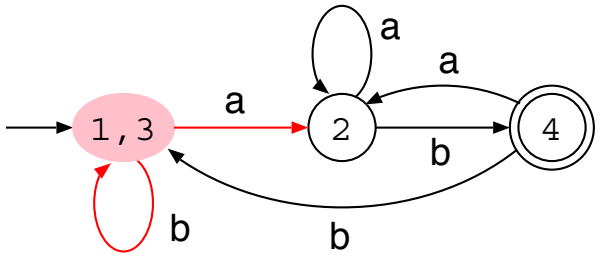
NFAからDFAへの変換





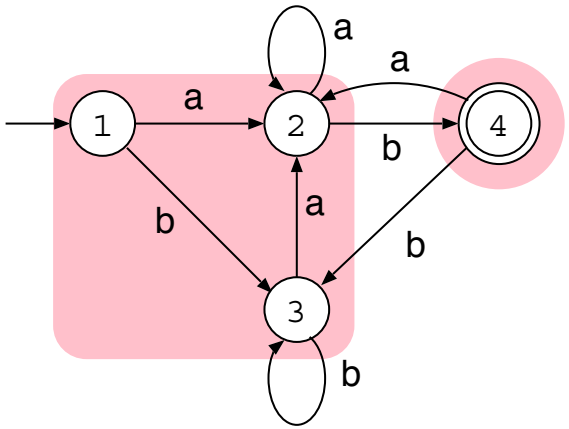
状態数最小の DFA

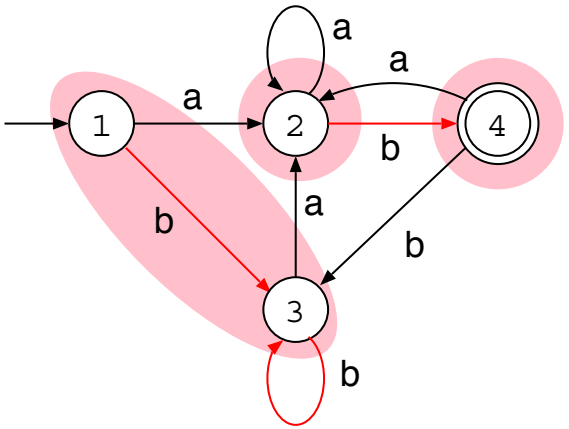




アルゴリズム

1. 受理状態とその他の 2 グループに分ける．
2. 同一文字に対する遷移先が異なる状態を，別グループに分ける．





字句解析プログラム

正規表現以外の規則

1. 最も長い字句要素を切り出す.
   例: $\mathtt{abc} = \mathtt{a} + \mathtt{bc} = \mathtt{ab} + \mathtt{c}$
2. 字句要素の種類ごとに優先度を設ける.
   例: `if` は識別子ではなく，キーワード

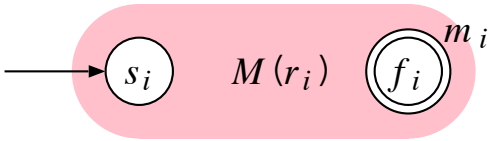
字句構造全体のDFA

$r_1, r_2, \ldots, r_n$: 字句要素の種類ごとの正規表現

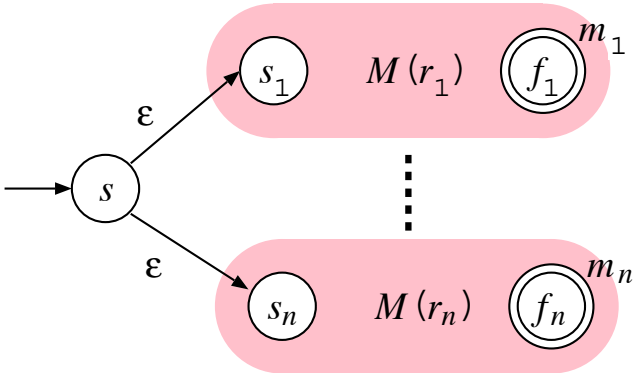
$r_1 | r_2 | \cdots | r_n$ のDFAでは種類がわからない

$m_1, m_2, \ldots, m_n$: 種類ごとのマーカ → 各受理状態に適切なマーカをつける

1. 種類ごとにNFA $M(r_i)$ を求め，受理状態にマーカをつける



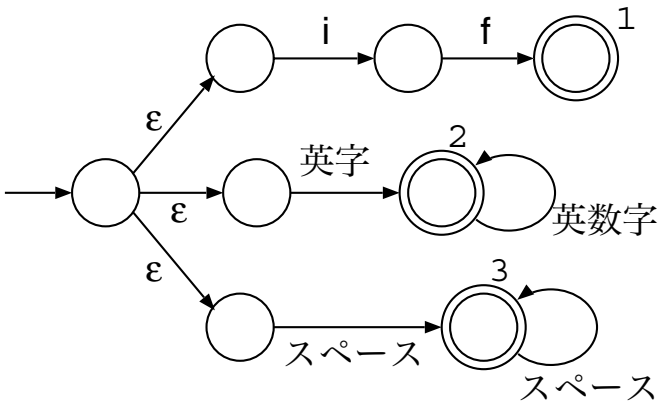
2. 新しい初期状態 s を用意し，全体のNFA M_{all} を作成.

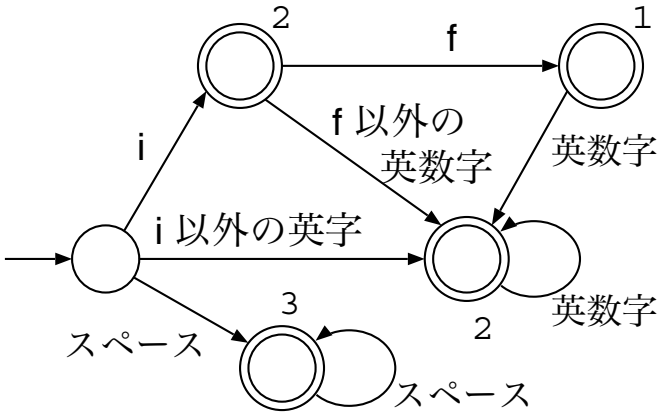


3. M_{all} をDFA $\widehat{M}_{all}$ に変換.
   個々の受理状態には，優先度最大のマーカを残す.
4. 状態数最小のDFAを作成.
   ただし，マーカの異なる受理状態は別グループ.

例

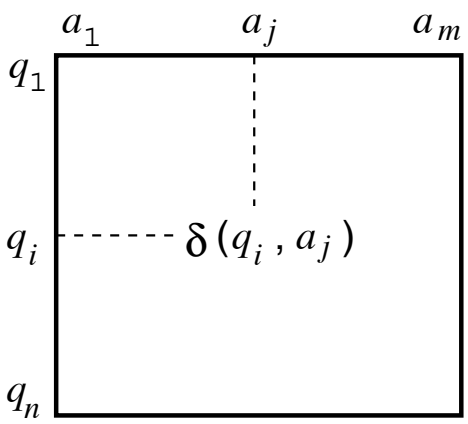
1. キーワード: `if`
2. 識別子: ⟨英字⟩⟨英数字⟩*
3. 空白: ⟨スペース⟩+





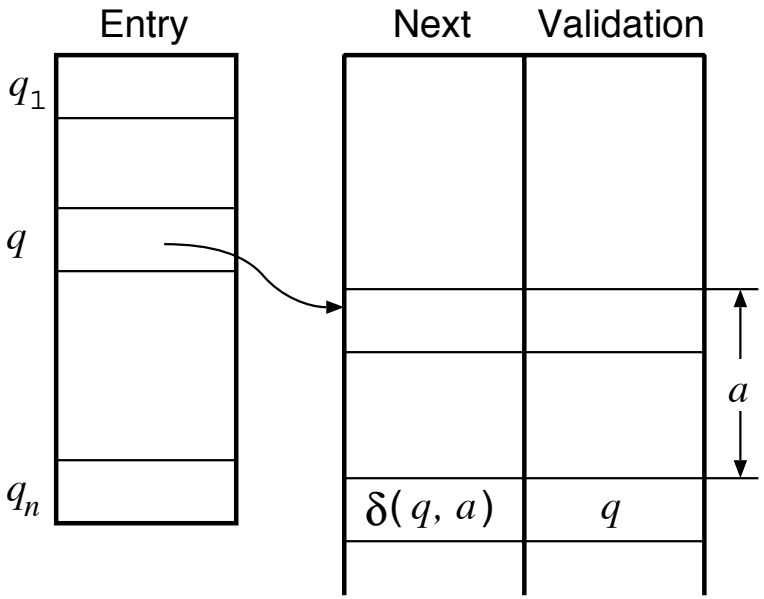
状態遷移表

状態遷移関数$\delta : Q \times \Sigma \to Q \cup \{\perp\}$を表にしたもの



ほとんどの要素が$\perp$（未定義）

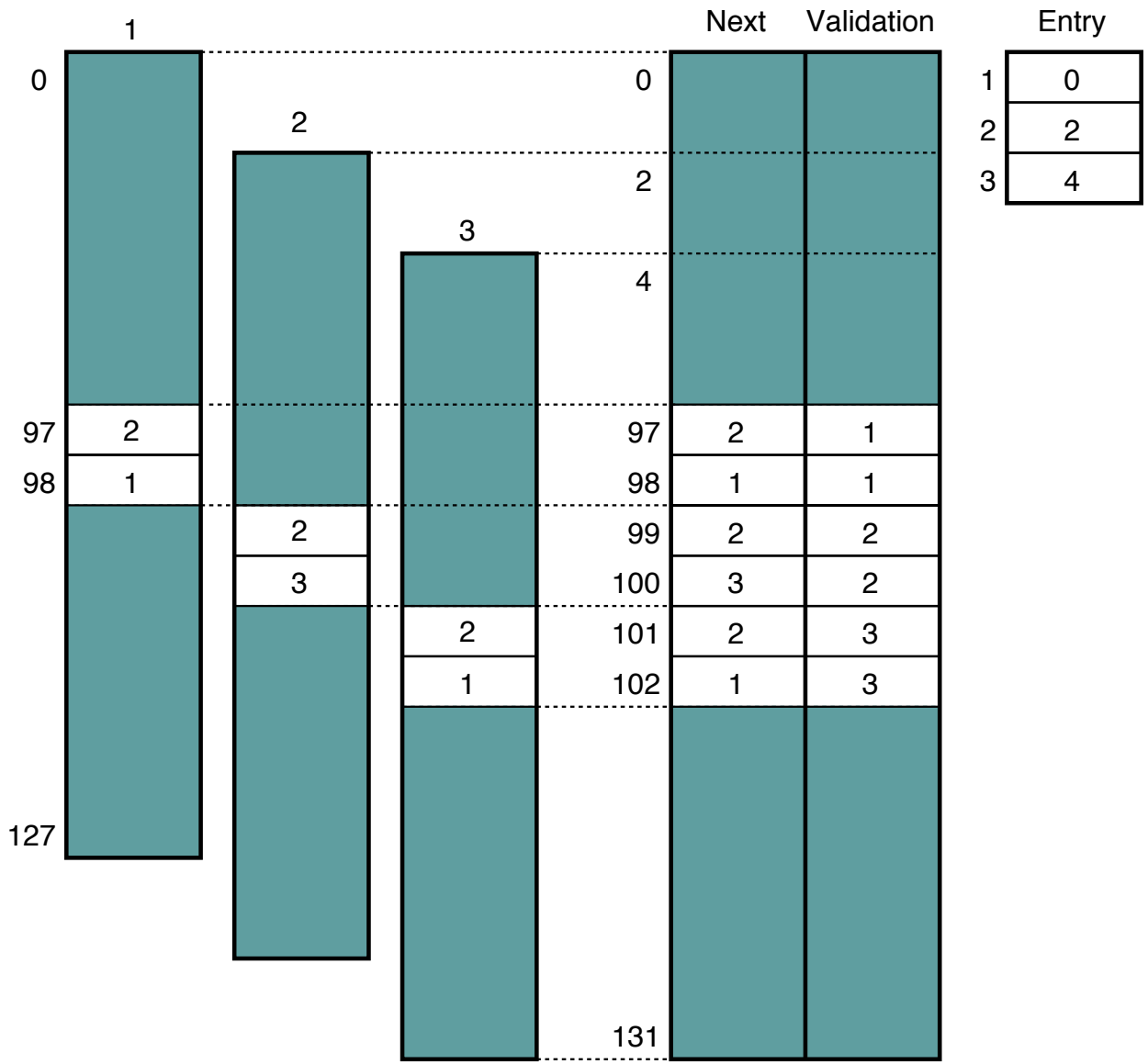
状態遷移表の圧縮

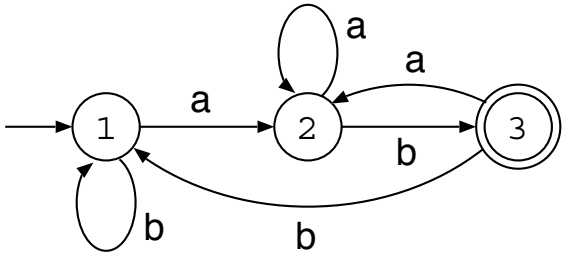


$$\texttt{Validation[Entry[}q\texttt{]+}a\texttt{]} = \begin{cases} q & (\delta(q,a) \neq \perp) \\ q \text{ 以外} & (\delta(q,a) = \perp) \end{cases}$$

```
next_state(q, a) {
  int i = Entry[q] + a;
  if (Validation[i] == q)
    return Next[i];
  else
    return UNDEFINED;
}
```

各ベクタの設定方法



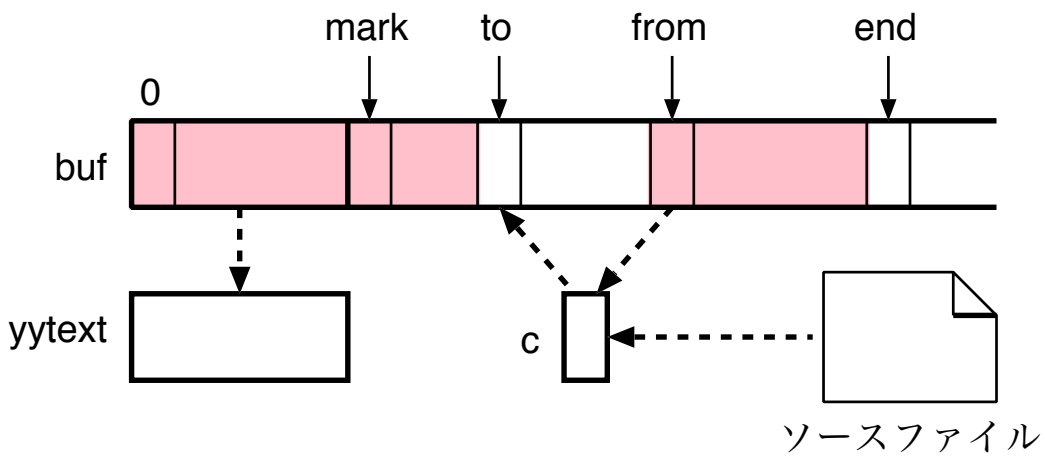


ASCII コードでは，’a’ = 97, ’b’ = 98

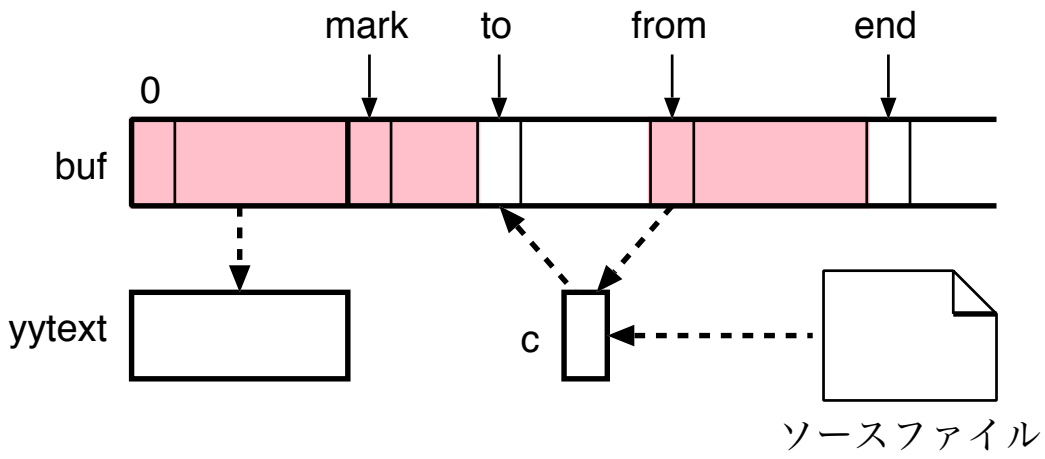
字句要素の切り出し

関数scan:

最も長い字句要素を切り出してyytextに格納し，マーカの値を返す．入力の終りに達していれば，ENDOFFILEを返す．



- buf[0]〜buf[to-1]:
  これまでに読み込んだ文字列
- buf[0]〜buf[mark-1]:
  最も長い字句要素
- buf[from]〜buf[end-1]:
  前回読み込みすぎた文字列



cへの文字設定

```
advance_c() {
  buf[to++] = c;
  if (from < end)
    c = buf[from++];
  else
    c = getc(in);
}
```

バッファの初期化

```
init_buf() {
  mark = to = from = end = 0;
  c = getc(in);
}
```

字句要素の切り出し

```
cutout_lexeme() {
  for (i = 0; i < mark; i++)
    yytext[i] = buf[i];
  yytext[i] = '\0';
  if (mark < to) {
    buf[to++] = c;
    while (from < end)
      buf[to++] = buf[from++];
    c = buf[mark++];
    from = mark;
    end = to;
  }
  mark = to = 0;
}
```

その他のサブ関数

`next_state`(q,a): 状態遷移関数$\delta(q, a)$の値

`final`(q): 状態qが受理状態かどうか

`marker`(q): 受理状態qのマーカ

`mark_buf()`: `to`の値を`mark`に保存

```
scan() {
  if (c == EOF) return ENDOFFILE;
  q = 1;  qf = UNDEFINED;
  for (;;) {
    if (final(q)) {
      mark_buf();  qf = q;
    }
    if ((x = next_state(q,c)) != UNDEFINED) {
      q = x;  advance_c();
    } else if (qf != UNDEFINED) {
      cutout_lexeme();  return marker(qf);
    } else {
      lexical_error();
      if (c == EOF) return ENDOFFILE;
      q = 1;
    }
  }
}
```

エラーリカバリ（次の空白まで読み飛ばす）

```
lexical_error() {
  while (c != ' ' && c != '\n' && c != EOF)
    advance_c();
  mark_buf();
  cutout_lexeme();
  printf("lexical error: '%s'\n", yytext);
}
```

字句解析の自動化

lex と flex

```
% cat scan.l
%%
[\t ]+ {}
(a|b)*ab { printf("%s: OK.\n", yytext);}
. { printf("%s: wrong.\n", yytext);}
%%
% lex scan.l
% cc lex.yy.c -ll
% a.out
aabbab
aabbab: OK.

aba
ab: OK.
a: wrong.

%
```

```
% cat scan1.l
%%
[\t ]+ {}
"+"|"-"|"*"|"/"|"="
    printf("%s (operator) ", yytext);
";"  printf("%s (separator) ", yytext);
[a-z][a-z0-9]*
    printf("%s (identifier) ", yytext);
[0-9\.]+ {printf("%s (number) ", yytext);}
.   printf("%s (don't know) ", yytext);
%%
% lex scan1.l
% cc lex.yy.c -ll
% a.out
abc=e*3.14+abc/e
abc (identifier) = (operator) e (identifier)
* (operator) 3.14 (number) + (operator)
abc (identifier) / (operator) e (identifier)
```

DFAから正規表現への変換

DFA $M = \langle \Sigma, \{q_1, q_2, \ldots, q_n\}, \delta, q_1, F \rangle$

R_{ij}^k: q_iからq_jへ，$q_1, q_2, \ldots, q_k$だけを通過して遷移させる文字列全体の集合

$$q_i \to \boxed{q_1 \sim q_k} \to q_j$$

1. $k > 0$のとき

$$R_{ij}^k = R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

$$q_i \to \boxed{q_1 \sim q_{k-1}} \to q_k \to \cdots \to q_k \to \boxed{q_1 \sim q_{k-1}} \to q_j$$

$$q_i \to \boxed{q_1 \sim q_{k-1}} \to q_j$$

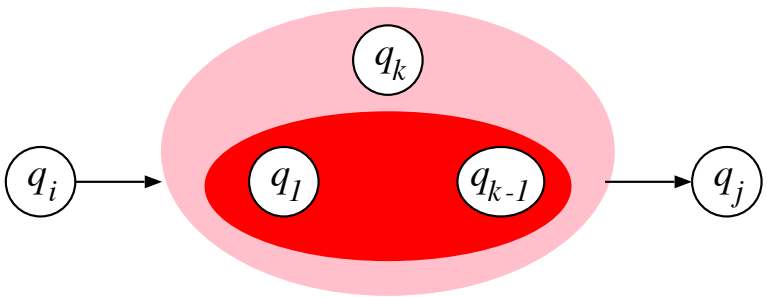
2. $k = 0$のとき

$$R_{ij}^0 = \begin{cases} \{a \mid \delta(q_i, a) = q_j\} & (i \neq j \text{のとき}) \\ \{a \mid \delta(q_i, a) = q_j\} \cup \{\varepsilon\} & (i = j \text{のとき}) \end{cases}$$

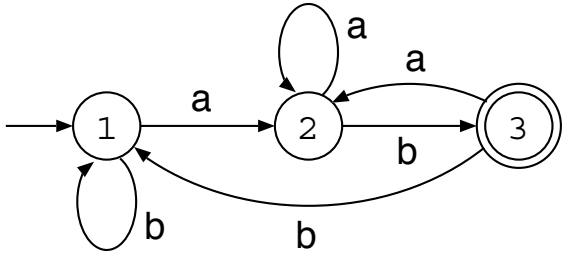
n: Mの状態数

$$L(M) = \bigcup_{q \in F} R_{sq}^n$$

この集合式を正規表現に変換する



例



$$R^3_{13} = R^2_{13}(R^2_{33})^* R^2_{33} \cup R^2_{13} = R^2_{13}(R^2_{33})^+ \cup R^2_{13} = R^2_{13}(R^2_{33})^*$$

$$R^2_{13} = R^1_{12}(R^1_{22})^* R^1_{23} \cup R^1_{13}$$

$$R^2_{33} = R^1_{32}(R^1_{22})^* R^1_{23} \cup R^1_{33}$$

$$R^1_{12} = R^0_{11}(R^0_{11})^* R^0_{12} \cup R^0_{12} = (R^0_{11})^+ R^0_{12} \cup R^0_{12}$$

$$= (R^0_{11})^* R^0_{12} = \{\mathtt{b}, \varepsilon\}^* \{\mathtt{a}\} = \{\mathtt{b}\}^* \{\mathtt{a}\}$$

$$R^1_{22} = R^0_{21}(R^0_{11})^* R^0_{12} \cup R^0_{22} = \phi \cup \{\mathtt{a}, \varepsilon\} = \{\mathtt{a}, \varepsilon\}$$

$$R^1_{23} = R^0_{21}(R^0_{11})^* R^0_{13} \cup R^0_{23} = \phi \cup \{\mathtt{b}\} = \{\mathtt{b}\}$$

$$R^1_{13} = R^0_{11}(R^0_{11})^* R^0_{13} \cup R^0_{13} = \phi$$

$$R^1_{32} = R^0_{31}(R^0_{11})^* R^0_{12} \cup R^0_{32} = \{\mathtt{b}\}\{\mathtt{b}\}^* \{\mathtt{a}\} \cup \{\mathtt{a}\} = \{\mathtt{b}\}^* \{\mathtt{a}\}$$

$$R^1_{33} = R^0_{31}(R^0_{11})^* R^0_{13} \cup R^0_{33} = \{\mathtt{b}\}\{\mathtt{b}\}^* \phi \cup \{\varepsilon\} = \{\varepsilon\}$$

$$R^3_{13} = R^2_{13}(R^2_{33})^*$$

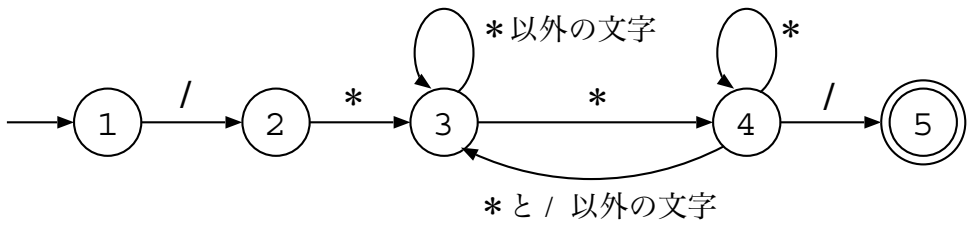
$$= (\{\mathtt{b}\}^* \{\mathtt{a}\}\{\mathtt{a}, \varepsilon\}^* \{\mathtt{b}\})(\{\mathtt{b}\}^* \{\mathtt{a}\}\{\mathtt{a}, \varepsilon\}^* \{\mathtt{b}\})^*$$

$$= (\{\mathtt{b}\}^* \{\mathtt{a}\}^+ \{\mathtt{b}\})(\{\mathtt{b}\}^* \{\mathtt{a}\}^+ \{\mathtt{b}\})^*$$

$$= (\{\mathtt{b}\}^* \{\mathtt{a}\}^+ \{\mathtt{b}\})^+$$

DFA と等価な正規表現は，$(\mathtt{b}^* \mathtt{a}^+ \mathtt{b})^+$

例：コメントの正規表現



$$/*\langle *\text{以外の文字}\rangle^* *$$
$$(\langle *\text{と}/\text{以外の文字}\rangle\langle *\text{以外の文字}\rangle^* * \mid *)^* /$$

状態番号の3と4を入れ替えると

$$/*\,(\langle *\text{以外の文字}\rangle \mid (*)^+\langle *\text{と}/\text{以外の文字}\rangle)^*\,(*)^*\,*/$$

正規表現の限界

正規表現では表せない文字列集合の例

$$X = \{a^i c b^i \mid i \geq 0\}$$

正規表現rで表せたと仮定する

n: $M(r)$の状態数

$a^n c b^n$ $(\in X)$を受理する$M(r)$の経路

$$q_0 \underbrace{\xrightarrow{a} \cdots \xrightarrow{a}}_{n} q_n \xrightarrow{c} q_{n+1} \underbrace{\xrightarrow{b} \cdots \xrightarrow{b}}_{n} q_{2n+1}$$

$q_i = q_j$ なる i, j $(0 \leq i < j \leq n)$が存在する

$$q_0 \underbrace{\xrightarrow{a} \cdots \xrightarrow{a}}_{i} q_i \underbrace{\xrightarrow{a} \cdots \xrightarrow{a}}_{j-i} q_j \underbrace{\xrightarrow{a} \cdots \xrightarrow{a}}_{n-j} q_n \xrightarrow{c} q_{n+1} \underbrace{\xrightarrow{b} \cdots \xrightarrow{b}}_{n} q_{2n+1}$$

q_iからq_jへの経路をカットすると

$$q_0 \underbrace{\xrightarrow{a} \cdots \xrightarrow{a}}_{i} q_i (= q_j) \underbrace{\xrightarrow{a} \cdots \xrightarrow{a}}_{n-j} q_n \xrightarrow{c} q_{n+1} \underbrace{\xrightarrow{b} \cdots \xrightarrow{b}}_{n} q_{2n+1}$$

$M(r)$は$a^{n-(j-i)} c b^n$ $(\notin X)$も受理する$\rightarrow$ 矛盾

文脈自由文法なら

$$A \rightarrow aAb \mid c$$