非決定性有限オートマトンの状態数最小化
Minimization of Nondeterministic Finite Automata

仙石 浩明　　矢島 脩三
Hiroaki SENGOKU　Shuzo YAJIMA

京都大学 工学部
Faculty of Engineering, Kyoto University

1 はじめに

非決定性有限オートマトン (NDFA) は形式言語理論の根幹を成し、符号理論においても重要な役割を果たしている[1, 3]。また、コンパイラなどの応用において重要な役割を果たしている[4]。従って、NDFAの性質を調べ理解することが重要である。

本報告では、NDFAの状態数を最小化する問題について考察し、最小化アルゴリズムを提案する。決定性有限オートマトン (DFA) は等価な状態を併合することによって唯一つの既約な DFA を作ることができ、さらに既約な DFA は同じ入力列を受理する DFA の集合の中で最小の状態数を持つことが知られている。ところが NDFA の場合は、一般に既約なものが状態数最小であるとは限らず、NDFAを最小化する効率的なアルゴリズムは知られていない。

NDFA は状態数が有限であるから、その状態数以下の全ての NDFA を探索すれば必ず状態数最小のものを見つけることが出来る。しかしこの方法は実際的ではない。最初にしらみつぶし的でないアルゴリズムが提案されたのは、Kameda, Weiner によってである[2]。このアルゴリズムでは与えられた NDFA が受理する入力列を逆順にしたものを受理する DFA と、与えられた NDFA と等価な DFA の二つの DFA を使って NDFA を正規化した Reduced Automaton Matrix を作る。この Matrix から NDFA を生成するのであるが、生成された NDFA の中には与えられた NDFA と等価でない場合がある。従って最小状態数の NDFA を探索する際、等価でない NDFA が生成された場合は別の NDFA を生成しなければならない。どのような場合に与えられた NDFA と等価でなくなるかは不明である。

本報告では、与えられた NDFA と同一の受理集合を持つ NDFA のみを探索するアルゴリズムを提案する。さらに状態数最小の NDFA の個数についても考察する。以下、2 章では基本的な諸定義を行う。3章では非決定性オートマトンを最小化する際に使う標準オートマトンについて述べる。4 章で最小化アルゴリズムを提案し、5章で考察を行う。

2 準備

定義 **1** (非決定性有限オートマトン) アルファベット Σ 上の非決定性有限オートマトン $\mathcal{A}$ を四つ組 $\mathcal{A} = (S, \delta, S_0, F)$ で表す。ただし、

- S : 状態の有限集合
- δ : 状態遷移関数 $\delta : S \times \Sigma \to 2^S$
- S_0 : 初期状態の集合 $S_0 \subset S$
- F : 受理状態の集合 $F \subset S$

遷移関数 δ の定義域を $2^S \times \Sigma$ に拡張する。すなわち $R \subset S, x \in \Sigma$ に対して

$$\delta(R, x) = \bigcup_{s \in R} \delta(s, x)$$

と定義する。さらに、定義域を $2^S \times \Sigma^*$ に拡張する。すなわち $\sigma = x_1 x_2 \cdots x_n$ に対し、

$$\delta(R, \sigma) = \delta(\delta(\ldots\delta(\delta(R, x_1), x_2), \ldots, x_{n-1}), x_n)$$

と定義する。

オートマトン $\mathcal{A}$ の受理入力列の集合を $\mathrm{bh}(\mathcal{A}) = \{\sigma \in \Sigma^* | \delta(S_0, \sigma) \cap F \neq \phi\}$ と定義する。また、状態 s からの受理入力列を $\mathrm{bh}(\mathcal{A}, s) = \{\sigma \in \Sigma^* | \delta(s, \sigma) \cap F \neq \phi\}$ と定義する。混乱の恐れがない限り、$\mathrm{bh}(\mathcal{A}, s)$ を $\mathrm{bh}(s)$ と略記する。

定義 **2** (等価性) $bh(\mathcal{A}) = bh(\mathcal{A}')$ が成り立つとき、そのときに限りオートマトン $\mathcal{A}$ と $\mathcal{A}'$ は等価であるとする。また、オートマトン $\mathcal{A}$ の状態 s, s' について、$bh(s) = bh(s')$ が成り立つとき、そのときに限り状態 s と s' は等価であるとする。

入力列 $\sigma = x_1 x_2 \cdots x_n$ を逆順にした列を $\overline{\sigma} = x_n \cdots x_2 x_1$ とする。この逆入力列を受理する逆オートマトンを定義する。

定義 **3** (逆オートマトン) $\mathcal{A}$ の逆オートマトンを $\overline{\mathcal{A}} = (S, \overline{\delta}, F, S_0)$ と定義する。ただし $\forall \sigma \in \Sigma^*, \forall s_i, s_j \in S$ に対して $s_i \in \overline{\delta}(s_j, \overline{\sigma}) \Leftrightarrow s_j \in \delta(s_i, \sigma)$。

非決定性オートマトンは次の subset construction によって、等価な決定性オートマトンに変換できる。

定義 **4 (subset construction)** 非決定性オートマトン $\mathcal{A} = (S, \delta, S_0, F)$ に対応する決定性オートマトンを $D(\mathcal{A}) = (P, \delta_P, P_0, F_P)$ と定義する。ただし、

- $P = \{\delta(S_0, \sigma) | \sigma \in \Sigma^*\} = \{p_1, p_2, \ldots p_m\}$
- $\delta_P(p_i, x) = \{\delta(p_i, x)\}$
- $P_0 = \{S_0\}$
- $F_P = \{p \in P | p \cap F \neq \phi\}$

$D(\mathcal{A})$ は等価な状態を持たないとする。すなわち任意の異なる状態 $p, p' \in P$ に対し、$bh(p) \neq bh(p')$.

3 標準非決定性オートマトン

決定性オートマトンの場合は、等価な状態の併合だけで最小状態数の等価なオートマトンに変換できる。ところが非決定性オートマトンの場合は、ある状態からの受理列の集合が他の 2 つの状態からの受理列の集合の和になる事がある。この場合は、その状態を分割し、それぞれに受理列が等しい状態を併合する事によって状態数が減る。

このような分割を行う必要がない、いわば「分割し尽くした」オートマトンがあれば、状態数の最小化を行う際便利であると思われる。そこで「分割し尽くした」オートマトンとして、標準非決定性オートマトンを定義する。

定義 **5** (標準非決定性有限オートマトン) 任意の異なる二状態 $s,s' \in S$ に対して $bh(s) \cap bh(s') = \phi$ が成り立ち、状態数が最小の非決定性有限オートマトン $\mathcal{A} = (S,\delta,S_0,F)$ を、標準非決定性有限オートマトン *(Normal Nondeterministic Finite Automaton)* と定義する。

標準オートマトンの状態を分割した場合、その状態からの受理列の集合はほかの状態からの受理集合と互いに素であるから、分割せずに併合操作を行った場合より分割を行ってから併合を行う場合の方が状態数が減るということは有り得ない。

定理 **1** 非決定性有限オートマトン $\mathcal{A} = (S,\delta,S_0,F)$ に対して、$\overline{\mathcal{A}}$ を *subset construction* により *DFA* 化し、状態数を最小化した決定性オートマトン $\mathcal{B} = D(\overline{\mathcal{A}})$ の逆オートマトン $\mathcal{C} = \overline{\mathcal{B}}$ は、$\mathcal{A}$ に等価な標準非決定性有限オートマトンである。

逆に、標準オートマトンの逆オートマトンは、状態数が最小の決定性オートマトンである。

証明: $\mathrm{bh}(s) \cap \mathrm{bh}(s') \neq \phi$ であるような $\mathcal{C}$ の状態 s,s' が存在したとする。$\sigma \in \mathrm{bh}(s) \cap \mathrm{bh}(s')$ と置く。

$\overline{\sigma}$ を C の逆オートマトンに入力すると s,s' に対応する状態に到達する。ところが、$\overline{\mathcal{C}} = \overline{\overline{\mathcal{B}}} = \mathcal{B}$ であり、$\mathcal{B}$ は決定性オートマトンなので $\overline{\sigma}$ を入力したとき異なる二つの状態に到達することはないので矛盾。故に $\mathcal{C}$ は標準オートマトンである。

逆は、明らかである。 証明終

たとえば次のような NDFA $\mathcal{A}$ を考える。

$\mathcal{A}$	0	1
$\rightarrow$ [1]	1	2
2	1	3
3	2	3

このとき逆オートマトン $\overline{\mathcal{A}}$ および DFA 化したオートマトンは次のようになる。

$\overline{\mathcal{A}}$	0	1
$\rightarrow$ [1]	1,2	–
2	3	1
3	–	2,3

$D(\overline{\mathcal{A}})$	0	1
$\rightarrow$ [1]	12	–
[12]	123	1
[123]	123	123

したがって標準非決定性オートマトン $\mathcal{C}$ は次のようになる。

$\mathcal{C}$	0	1
$\rightarrow$ [1]	–	12
$\rightarrow$ 12	1	–
$\rightarrow$ 123	12,123	123

標準 NDFA の状態を適当に併合することにより、標準オートマトンに等価な状態数が最小の NDFA を構成する事が出来る。以下、標準 NDFA から状態数最小の NDFA を構成する方法について考察する。

標準 NDFA を $\mathcal{N} = (S_N,\delta_N,N_0,F_N)$, (ただし $S_N = \{n_1,n_2,\ldots,n_p\}$) とする。これに等価な NDFA は一般に $\mathcal{A} = (S,\delta,S_0,F)$, $S = \{s_1,s_2,\ldots,s_m\}$ と表すことが出来る。ただし、

- $S \subset 2^{S_N}$ すなわち、$s_i \in S$ は S_N の部分集合。
- $\delta(s_i,x) = \mathrm{Map}(\bigcup_{n_i \in s_i} \delta_N(n_i,x))$
  $= \mathrm{Map}(\delta_N(s_i,x))$
- $S_0 = \mathrm{Map}(N_0)$
- $F = \{s_i | s_i \cap F_N \neq \phi\}$

ここで $\mathrm{Map}(R_N)$ $(R_N \subset S_N)$ は、$\cup\mathrm{Map}(R_N) = R_N$ を満たすような、S の部分集合である。つまり、関数 $\mathrm{Map} : 2^{S_N} \rightarrow 2^S$ が存在するような S を決める必要がある。なお、$\mathrm{Map}(R_N) = \{s_i | s_i \subset R_N\}$ とすれば、常に $\cup\mathrm{Map}(R_N) \subseteq R_N$ となる。故に任意の S において、$\mathcal{A}$ の受理集合 $\mathrm{bh}(\mathcal{A})$ は $\mathcal{N}$ の受理集合 $\mathrm{bh}(\mathcal{N})$ に含まれるように出来る。

4 最小化アルゴリズム

標準 NDFA $\mathcal{N} = (S_N,\delta_N,N_0,F_N)$ から、この NDFA と等価で、状態数が最小の NDFA $\mathcal{A} = (S,\delta,S_0,F)$ を構成する方法について考える。

任意の S_N の部分集合の集合 $S \subset 2^{S_N}$ のうち、Map が存在するもので、状態数が最小のものを見つければ良い。

定理 **2** 標準 *NDFA* $\mathcal{N} = (S_N,\delta_N,N_0,F_N)$ から生成された等価な *NDFA* $\mathcal{A} = (S,\delta,S_0,F)$ の任意の状態遷移 $s_j \in \delta(s_i,x)$ に対し、次の条件が成立する。

$$\bigcup_{n_j \in s_j} \overline{\delta_N}(n_j,x) \subset s_i$$

証明: この条件が成立しない遷移 $s_j \in \delta(s_i,x)$ が存在したとする。すると

$$n_i \in \bigcup_{n_j \in s_j} \overline{\delta_N}(n_j,x) \backslash s_i$$

となる $n_i \in S_N$ が存在する。n_i は標準 NDFA の状態であるから、$n_i \notin s_i$ より $\mathrm{bh}(n_i) \cap \mathrm{bh}(s_i) = \phi$。ところが、$\delta_N(n_i,x) \subset s_j$ かつ $s_j \in \delta(s_i,x)$ すなわち $s_j \subset \delta_N(s_i,x)$ であるから、

$$\mathrm{bh}(\delta_N(n_i,x)) \subset \mathrm{bh}(\delta_N(s_i,x))$$

故に矛盾。 証明終

定理2 を使うと、$S \subset 2^{S_N}$ の探索時に枝刈りすることが可能になる。すなわち S の要素を一つづつ見つけていく過程で、この定理の条件を満たさない物を除去することが出来る。

たとえば、標準オートマトン $\mathcal{N} = (S_N,\delta_N,N_0,F_N)$, $S_N = \{0,1,2,3,4,5\}$ が次のように与えられたとする。

$\mathcal{N}$	0	1
[0]	–	–
1	0	4
$\rightarrow$ 2	1,2,3,5	2,3
3	–	0,5
$\rightarrow$ 4	–	1
$\rightarrow$ 5	4	–

最初、$S = \phi$ とする。$N_0 = \{2,4,5\}$, $S_0 = \mathrm{Map}(N_0)$ であるから、S_0 の候補として次の 5 通りが考えられる。

1. $\{\{2,4,5\}\}$
2. $\{\{2,4\},\{5\}\}$
3. $\{\{2,5\},\{4\}\}$
4. $\{\{4,5\},\{2\}\}$
5. $\{\{2\},\{4\},\{5\}\}$

1 番目の候補を選ぶと、$S_0 = \{s_1\}$ $(s_1 = \{2,4,5\})$ となる。この s_1 を S に追加する。従って $S = \{s_1\}$。

次に s_1 からの遷移について考える。まず、$\delta(s_1,0)$ について、

$$\delta(s_1,0) = \mathrm{Map}(\delta_N(s_1,0))$$

$$\delta_N(s_1,0) = \{1,2,3,4,5\}$$

だから、$\exists R \subset S, \cup R = \{1,2,3,4,5\}$ が成り立つようにS に要素を追加しなければならない。$\{1,2,3,4,5\} \backslash s_1 = \{1,3\}$ であるから、S に追加する集合は、1,3 の両方を含む集合一つか、1 を含む集合と 2 を含む集合一つづつになる。この時これらの集合は 0 を含む事はない。なぜなら、0 を含むと仮定すると、定理2 より $s_1 \supset \overline{\delta_N}(0,0) = \{1\}$ となり、矛盾。

定義 **6 (Cover)** $S \subset 2^{S_N}$, $R_{N1}, R_{N2} \subset S_N$ が与えられたとき、次の条件を全て満たす集合 R 全てからなる集合を $Cover(S, R_{N1}, R_{N2})$ と定義する。

- $R_{N1} \subset \cup R \subset S_N \backslash R_{N2}$
- $\forall s \in R, R_{N1} \not\subset \cup(R \backslash \{s\})$
- $\forall s \in R, \forall s' \in S,\ s' \not\subset s$

ただし、$R_{N1} = \phi$ の場合は $Cover(S, \phi, R_{N2}) = \{\phi\}$ とする。

つまり、$\mathrm{Cover}(S, R_{N1}, R_{N2})$ は、R_{N2} を含まずに R_{N1} を覆い尽すために必要な S に追加すべき状態の集合を列挙したものである。この記法を用いると、$\delta(s_1,0)$ について考えたとき S に追加すべき状態の集合は、$R \in \mathrm{Cover}(S, \{1,3\}, \{0,4,5\})$ と書ける。

$\delta(s_1,1)$ についても同様にして、S に追加する状態の集合は、$\mathrm{Cover}(S, \{1,2,3\}, \{0,4,5\})$ の中から選べば良い。

$$\{s_2\} \in \mathrm{Cover}(S, \{1,3\}, \{0\}),$$

$$\{s_2\} \in \mathrm{Cover}(S, \{1,2,3\}, \{0,4,5\})$$

(ただし $s_2 = \{1,2,3\}$) を選ぶと、$\delta(s_1,0) = \{s_1, s_2\}$, $\delta(s_1,1) = \{s_2\}$ となる。従って $S = \{s_1, s_2\}$ とする。

s_2 からの遷移を考えると、次に S に追加する状態の集合は、$\mathrm{Cover}(S, \{0,5\}, \{4\})$, $\mathrm{Cover}(S, \{0,3\}, \{1\})$ の中から選べば良い。$\{s_3\}$ (ただし $s_3 = \{0,2,3,5\}$) を選ぶことが出来て、$S = \{s_1, s_2, s_3\}$ となる。

$\delta(s_3,0) = \{s_1, s_2\}$, $\delta(s_3,1) = \{s_3\}$ であるから、S にこれ以上状態を追加する必要は無い。

同様にして、ほかの選択枝を探索すると、状態数 3 未満の解が無いことがわかる。したがって、次の状態数最小の非決定性オートマトンを得ることが出来る。

	0	1
$\rightarrow s_1$	s_1, s_2	s_2
s_2	s_2, s_3	s_1, s_3
$\boxed{s_3}$	s_1, s_2	s_3

最小化アルゴリズムをまとめると次のようになる。

procedure Minimize($\mathcal{N} = (S_N, \delta_N, N_0, F_N)$)
  $m \leftarrow \infty$
  $S_{min} \leftarrow$**Undefined**
  for all $R \in \mathrm{Cover}(\phi, N_0, S_N \backslash N_0)$
    Search(R, R)
  return S_{min}
end

ただし、Search(S,T) は、探索中のオートマトンの状態集合が S であり、$T \subset S$ に含まれる状態からの状態遷移がまだ探索されていない時、残りの状態遷移について探索を行う。

Procedure Search(S,T)
  if $T = \phi$ **then begin**
    if $|S| < m$ **then begin**
      $S_{min} \leftarrow S$
      $m \leftarrow |S|$
    end
    return
  end
  $s \in T$
  $T \leftarrow T \backslash \{s\}$
  for all $x \in \Sigma$ **begin**
    $R_N \leftarrow \bigcup_{n_i \in s} \delta_N(n_i, x)$
    $R_{N2} \leftarrow S_N \backslash R_N$
    $R' \leftarrow \{s_i \in S | s_i \subset R_N\}$
    $R_{N1} \leftarrow R_N \backslash \cup R'$
    for all $R \in \mathrm{Cover}(S, R_{N1}, R_{N2})$
      Search($S \cup R$, $T \cup R$)
  end
end

5 状態数最小の NDFA の個数

定義 **7 (Reduced Automaton Matrix)** 標準 *NDFA* $\mathcal{N} = (S_N, \delta_N, N_0, F_N)$ *($S_N = \{n_1, n_2, \ldots, n_q\}$)* が与えられたとき、$\mathcal{N}$ から *Subset Construction* によって構成した *DFA* を $\mathcal{M} = D(\mathcal{N}) = (S_M, \delta_M, M_0, F_M)$ *($S_M = \{m_1, m_2, \ldots, m_p\}$)* とする。この時次の $p \times q$ 行列を *Reduced Automaton Matrix[2]* と呼ぶ。

$$(a_{ij}), \qquad a_{ij} = \begin{cases} 1 & (n_j \in m_i \text{ の時}) \\ 0 & (\text{それ以外}) \end{cases}$$

ただし、等しい列ベクトル/行ベクトルは取り除く。

Reduced Automaton Matrix 上に grid[2] を定義する。

定義 **8 (grid)** *Reduced Automaton Matrix* の行 $m_{i_1}, \ldots, m_{i_a}$ と列 $n_{j_1}, \ldots, n_{j_b}$ の全ての交点が *1* である時、$g = \{m_{i_1}, \ldots, m_{i_a}; n_{j_1}, \ldots, n_{j_b}\}$ は *grid* であるという。

標準 NDFA $\mathcal{N} = (S_N, \delta_N, N_0, F_N)$ から構成した等価な NDFA $\mathcal{A} = (S, \delta, S_0, F)$ の状態 $s_i \in S$ について、$\delta(s_i, x) = \mathrm{Map}(\delta_N(s_i, x))$ が成り立つ。これを繰り返し用いると、任意の入力列 $\sigma \in \Sigma^*$ に対して、

$$\delta(S_0, \sigma) = \mathrm{Map}(\delta_N(S_0, \sigma))$$

が成立する。Reduced Automaton Matrix 上の全ての 1 (つまり $\forall n_j, m_i, n_j \in m_i$) に対して、$\exists s \in \mathrm{Map}(m_i) \wedge n_j \in s$。

また、各 $s \in S$ に対して $s \in \mathrm{Map}(m_i)$ を満たす m_i の集合と、$n_j \in s$ を満たす n_j の集合、すなわち grid が対応する。故に Reduced Automaton Matrix 上の全ての 1 は $s \in S$ に対応する grid で覆い尽くされる。

さて、ここで $\mathcal{A}$ が $\mathcal{N}$ に等価な全ての NDFA の中で状態数が最小であるとする。$\mathcal{A}$ の状態数を m とする。すると、$\mathcal{N}$ の Reduced Automaton Matrix は m 個の grid で覆い尽くされる。m 個の grid で覆い尽くされる Reduced Automaton Matrix のうち最大のものは、各行及び列の 1 を含む grid の組み合わせが全ての場合を尽くすものである。例えば $m = 3$ の時は次のようになる。

$$X_m = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Reduced Automaton Matrix が X_m になる NDFA は、入力アルファベットの個数が十分大きければ常に構成可能である。

定義 9 (拡大 **Reduced Automaton Matrix**) m 個の *grid*

$$Xg_j = \{ \begin{array}{l} i\text{行} \mid i \text{を } 2 \text{進表示したとき } j \text{ 桁目が } 1 \\ i\text{列} \mid i \text{を } 2 \text{進表示したとき } j \text{ 桁目が } 1 \end{array} \begin{array}{l} ; \\ \} \end{array}$$

$$(j = 1, 2, \ldots, n)$$

で覆い尽くす事が出来る $2^m - 1$ 次の正方行列 X_m を拡大 *Reduced Automaton Matrix (Expanded Reduced Automaton Matrix)* と定義する。

拡大 Reduced Automaton Matrix は、各行及び列の 1 を含む grid の組み合わせが全ての場合を尽くす ($2^m - 1$ 通り) なので、m 個の grid で覆い尽くす事が出来る全ての Reduced Automaton Matrix を小行列として含む。

例えば、4章で最小化したオートマトンの場合、Reduced Automaton Matrix は次のようになる。

$\mathcal{M}$ \ $\mathcal{N}$	0	1	2	3	4	5
0	0	0	1	0	1	1
1	0	1	1	1	1	1
2	1	1	1	1	1	1
3	0	1	1	1	0	0
4	1	1	1	1	0	1
5	1	0	1	1	1	1

この Matrix は、X_3 の行を 2,6,7,4,5,3 行の順に並べ替え、列を 1,4,7,5,2,3 列の順に並べ替え、1 行と 6 列を除くことによって得ることが出来る。この時、X_3 の grid $Xg_3 = \{4,5,6,7\text{行}; 4,5,6,7\text{列}\}$ は、状態数最小のオートマトン ($S = \{s_1, s_2, s_3\}$) の状態 s_2 に対応し、$Xg_2 = \{2,3,6,7\text{行}; 2,3,6,7\text{行}\}$ は状態 s_1 に、$Xg_1 = \{1,3,5,7\text{行}; 1,3,5,7\text{列}\}$ は状態 s_3 に、それぞれ対応する。

この様に、最小化したときの状態数が m になるオートマトンの Reduced Automaton Matrix は全て X_m に小行列として含まれる。逆に言うと、与えられたオートマトンの Reduced Automaton Matrix が X_m に含まれるどの小行列に対応するかが決まれば、最小状態数のオートマトンは一意に決定される。このことから X_m に小行列として含まれる Reduced Automaton Matrix の個数 (対称性を除外する) は、状態数最小のオートマトンの個数の上界になる。

6 おわりに

NDFA の状態数の最小化を行うアルゴリズムを提案した。Reduced Automaton Matrix を用いる方法 [2] ではどの様なときに与えられた NDFA と等価でない NDFA が構成されるかが不明であったが、標準 NDFA を考えることにより明確にした。すなわち、定理 2 を満たさない遷移がある場合に等価でなくなる。さらに従来法 [2] の探索において、定理 2 を用いて枝刈りを行う事により、高速化する事も可能である。

また拡大 Reduced Automaton Matrix を使って、状態数最小の NDFA の個数の上界を求める方法を提案した。状態数最小の NDFA の個数は多くの場合 1 と予想されるが、どのような場合に 1 となるかは今後の課題である。

謝辞

有益なご助言、ご討論頂く矢島研究室の皆様に感謝いたします。

参考文献

[1] Z. Kohavi. *"Switching and Finite Automata Theory."* Teta McGraw-Hill, 1970

[2] T. Kameda and P. Weiner. *"On the State Minimization of Nondeterministic Finite Automata"* IEEE Trans. Computer vol C-19, No. 7, July 1970

[3] A. Restivo. *"Codes and Automata"* Lecture Notes in Computer Science vol 386, May 1988

[4] 加藤, 稲垣, 本多. 「有限オートマトンの合成アルゴリズムの改良とその字句解析プログラム自動生成系への応用」 電子通信学会技術研究報告 AL83-27