オートマトン機構の染色体表現とその応用

藤野　精一*・矢倉　昌和**

On A Chromosome Representation of Finite Automaton and Its Application

Seiichi FUJINO and Masakazu YAKURA

Abstract

In this paper we shall define a chromosome representation of finite automaton and study its application. It will be easy to construct a program in which strings, randomly selected, can be recognized by making use of the representation. By using such a programming technique we can estimate the degree of approximation of a finite automaton with respect to a given finite automaton.

1．はじめに

機構が未知の有限オートマトンに対して，そのオートマトンで受理される記号列を有限個と，受理されない記号列を有限個与えてその有限オートマトンの機構を推定するという問題は有限オートマトンが研究テーマになった当初から例による推定（inference by example）とよばれて研究の主要テーマの１つとして盛んに研究された[1]。しかし，若干の進歩はあったが，決定的な結果は未だ得られていない。

この論文では有限オートマトンの染色体表現というものを定義し，その表現にできるかぎり近づけるような方法を提案したい。いいかえると，有限オートマトンを１意に表現する表現法として或る表現を定義する。この定義自体はまったく新しいアイデアではない。すでに[２]において和田が使用している。この和田の考え方を進めて任意の有限オートマトンが１意に表現できるように改良を施したものである。その表現が，与えられたオートマトンを１つの個体とみたとき，その個体の染色体ともみなされることから染色体とよぶことにしたのである。その染色体表現を利用すると無作為に選ばれた記号列がそのオートマトンで受理されるか，受理されないか容易に判定されるプログラムを作成することができる。

元の有限オートマトンの機構は未知であるがそのオートマトンが受理する記号列，受理しない記号列が有限個わかっているとしたとき，無作為に染色体表現を作成し，その染色体表現をもつオートマトンで先のオートマトンが受理した記号列をどの程度受理するか，また受理しなかった記号列をどの程度受理しないかをプログラムで求めることができればそのオートマトンの機構が元のオートマトンの機構に対してどの程度の近似度をもっているか計算することができる。あとは染色体を適当に変更してその近似度をどのようにあげていくかが問題になる。

このようなことをC言語でプログラムを作成して考えることにする。プログラミングに際して［３］のＳＣＬ（Satoshi C Libraries）を利用してプログラム作成を容易にした。

2．オートマトンとその表現

Xを有限個の記号からなる集合とする。これをアルファベットという。アルファベットXの上の有限オートマトンとは次の集合と写像とからなるシステムAをいう。

$A=\langle S, X, f, s_0, D\rangle$

ここにS，Xはそれぞれ状態集合，アルファベットと呼ばれる有限集合で，Sの要素はそれぞれ状態とよばれる。写像fは機構とよばれる$S\times X$からSへの写像で，$s_0(\in S)$は初期状態，Dは最終状態と呼ばれる（Sの）部分集合である。この写像と集合の組で１つの有限オートマト

＊教養部　　＊＊大学院工学研究科
平成８年９月４日受理

ンが構成される。

例1． 次のシステムを考えよう。

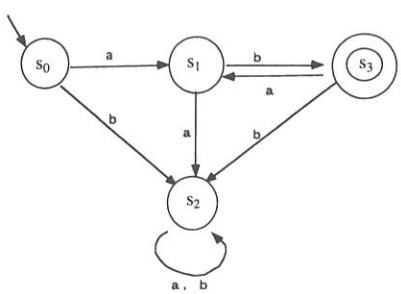
$A=\langle S, X, f, s_0, D\rangle$

ここに $S=\{s_0, s_1, s_2, s_3\}$, $X=\{a, b\}$ である。f は $S\times X$ から S への写像で次の機構表で表される（表－1）。この機構表はすべての i，j について状態 $f(s_i, x_j)$($i=0, 1, 2, 3$; $j=0, 1$ ただし $x_0=a$, $x_1=b$ とする。）を状態 s_i の行と入力 x_j の列の交点に書いて作成する表である。

表－1　機　構　表

状態＼入力	a	b
s_0	s_1	s_2
s_1	s_2	s_3
s_2	s_2	s_2
s_3	s_1	s_2

例えば現在の状態が s_1 で，読み取った入力記号が b のとき次の状態は s_3 である。$f(s_1, b)=s_3$ を示している。状態を丸印で，状態の遷移を，読み取る入力記号を伴った矢印で表し，矢印の始点は現在の状態，矢印の先は次の状態としてすべての状態について遷移の矢印を書くと次のような視覚的に見やすい（**状態図**と呼ばれる）有向グラフが得られる（図－1）。



図－1　状　態　図

このとき，初期状態と最終状態の集合とを図のなかに明示するため，初期状態 s_0 には出発点を示す矢印を，最終状態には2重丸印を付けてそれぞれを区別する。上の場合では $D=\{s_3\}$ である。このときシステム A は4個の状態からなる状態集合，アルファベット $\{a, b\}$，初期状態 s_0，最終状態 $D=\{s_3\}$ の有限オートマトンである。

機構 f は $S\times X^*$ から S への写像 f^* に次のように拡張される。ここに X^* は X の記号を有限個並べてできる記号列の全体で，$u, v, w, \ldots$ のような英小文字の最後の方の文字で X^* の要素を表す。とくに λ で長さ0の記号列を示す。

$$\begin{cases} f^*(s, \lambda)=s(s\in S), \\ f^*(s, wx)=f(f^*(s, w), x)\ (s\in S, w\in X^*, x\in X) \end{cases}$$

この f^* は状態 s で入力記号列 w を読みこむとき，$f^*(s, w)$ で最後に移った状態を示す。拡張された写像 f^* を用いて記号列の集合 $\{w|w\in X^*, f^*(s_0, w)\in D\}$，すなわち，初期状態 s_0 で入力記号列の各記号をつぎつぎと読み込むとき，読み終った最後の状態が D にはいる記号列 w の全体が定まる。この集合はオートマトン A によって一意に定まる集合であるから，これを $L(A)$ と書き，オートマトン A によって受理される言語という。

いま，記号列 w が $L(A)$ に入るときこの w を $L(A)$ の＋サンプル，入らないとき－サンプルとよぶ。

例2． 例1のオートマトン A によって受理される言語 $L(A)$ は次の（記号列の）集合である。

$L(A)=\{(ab)^n|n=1, 2....\}=\{ab, abab, ababab, ...\}$

したがって，たとえば，$abab$ は＋サンプル，$aabba$ は－サンプルである。

3．有限オートマトンの染色体表現

X の上のオートマトン $A=\langle S, X, f, s_0, D\rangle$ に対して $Z_n=\{0, 1, 2...., n-1\}$ 上のオートマトン $A^\#=\langle Z_m, Z_n, f^\#, 0, D^\#\rangle$ を次のように作成する。ここで $S=\{s_0, s_1...., s_{m-1}\}$，$X=\{x_0, x_1,....x_{n-1}\}$ とする。写像 $\#_1: S\to Z_m$, $\#_2: X\to Z_n$, $\#_3: S\times X\to Z_m\times Z_n$ を

$\#_1(s_i)=i\ (s_i\in S, i=0, 1, 2,...., m-1)$,

$\#_2(s_j)=j\ (x_j\in X, j=0, 1, 2,...., n-1)$,

及び

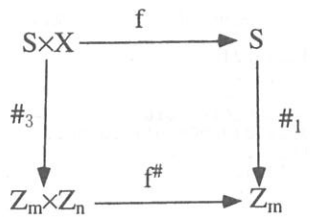
$\#_3(s_l, x_j)=(i, j)\ (s_l\in S, x_j\in X; i=0, 1, 2,...., m-1, j=0, 1, 2,...., n-1)$ で定義する。すなわち，S，X の各要素に一定の順序で番号をつける。その作用を写像 $\#_1$，$\#_2$ で表現する。集合 $S\times X$ の各要素 (s, x) には番号の組 $(\#_1(s), \#_2(x))$ を対応させる。

$\#_3(s_l, x_j)=(\#_1(s_l), \#_2(x_j))\ (i=0, 1,...., m-1, j=0, 1,...., n-1)$ である。このとき機構 f に $Z_m\times Z_n$ 上で対応させる写像が $f^\#$ である。

$f: S\times X\to S$ に対して $f^\#: Z_m\times Z_n\to Z_m$ を $\#_1 f=f^\#\#_3$，即ち，次図を可換にするように定める（図－2）。

$f^\#$ は $f(s_l, x_j)=s_K$ のとき $f^\#(i, j)=k(s_l, s_K\in S, x_j\in X)$ と定義すればよい。

この対応により任意の $s\in S$, $x\in X$ に対して



図－2　可　換　図

$\#_1(f(s, x))=f^{\#}(\#_3(s, x))$,

すなわち，図－2は可換である。

また集合 $D^{\#}$ は Z_m の部分集合で $\#_1D$ である。すなわち，$D=\{d_0, d_1.....d_{p-1}\}(\subseteq S)$ とするとき $D^{\#}=\{\#_1(d_0), \#_1(s_1),....\#_1(d_{p-1})\}$ である。

このとき非負整数よりなる次の記号列をオートマトン A の**染色体**とよび記号 $g(A)$ と書く。

$g(A)=mnpg_0g_1....g_{mn-1}g'_0g'_1.....g'_{p-1}$,

ただし，$g_{in+j}=\#_1(f(s_i, x_j))$ $(i=0, 1,...., m-1; j=0,1,...., n-1)$,

$g'_k=\#_1(d_k)$ $(k=0, 1, ..., p-1)$

である。$g(A)$ を構成する記号列は左から順に

m：状態の総数

n：入力記号の総数

p：最終状態の総数

g_{in+j}：$f(s_i, x_j)$に対応する数字

$(i=0, 1,....,m-1; j=0, 1,....,n-1)$

表－2　$\#_1$

S	$\#_1(S)$
s_0	0
s_1	1
s_2	2
s_3	3

表－3　$\#_2$

X	$\#_2(X)$
a	0
b	1

表－4　$\#_3$

$S\times X$	$\#_3(S\times X)$
(s_0, a)	(0, 0)
(s_0, b)	(0, 1)
(s_1, a)	(1, 0)
(s_1, b)	(1, 1)
(s_2, a)	(2, 0)
(s_2, b)	(2, 1)
(s_3, a)	(3, 0)
(s_3, b)	(3, 1)

g'_k：d_kに対応する数字$(k=0, 1,...., p-1)$

である。

例3. 例1のオートマトンAの染色体$g(A)$は次のようになる。

$g(A)=421122322123$

ここに$\#_1$, $\#_2$および$\#_3$は次表で与えられる（表－2，3，4）。

染色体 $g(A)$ は図－3で示すように矢印の始点に指示した数に対応する各数字を羅列する。

4．非受理記号列の判定プログラム

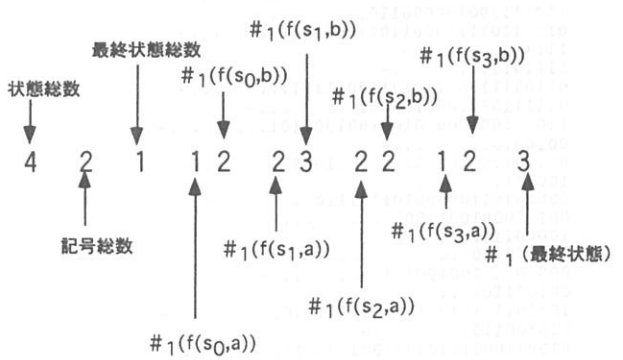
有限オートマトンの染色体表現はまず第一にそのオートマトンによって受理される記号列や受理されない記号列を判定するプログラムの作成に応用できる。

次のプログラム（表－5）は有限オートマトンを図－3の染色体で表現されるオートマトンにとるとき，長さ30以内の記号列を30語ランダムに作成し各記号列がこのオートマトンで受理されるとき＋を，受理されないとき－をプリントし，最後に＋の記号列の総数を印刷するプログラムである。

プログラムの主要部分は

```
s=0;
for (j=0; j<1; j++)
{
x=w[j];
t=n*s+x+3;
s=a[t];
}
```

の部分である。配列 w を読み込み，t で $f^*(s_0, w[0]w[1]...w[j])$の入っている位置を計算し $a[t]$ でそのときの状態，すなわち，$f^*(s_0, w[0]w[1]...w[j])$ を求め



図－3　染色体の構成

表— 5　記号列判定プログラム　(1)

```
#include  <SCL.h>
#include  <stdio.h>
#include  <stdlib.h>

main()
{
    int  a[12]={4,2,1,1,2,2,3,2,2,1,2,3},s,d=3,n=2;
    int  w[30][30];
    int  i,j,t,x,l,c=0;
    StartTextOnly();
    BeginTextPrinting();
    printf("a={");
        for(i=0;i<12;i++)
            printf("%d",a[i]);
            printf("}\n");

    for(i=0;i<30;i++)
    {
        s=0;
        srand(TickCount());
        l=(double)rand()/32767*29+1;
        for(j=0;j<l;j++)
        w[i][j]=(int)rand()/16390;
        for(j=0;j<l;j++)
        {
            x=w[i][j];
            t=n*s+x+3;
            s=a[t];
        }

        for(j=0;j<l;j++)
            printf("%d",w[i][j]);
        if(s==d)
        {
            printf("...   ... +\n");
            c++;
        }
        else
        {
            printf("...    ...-\n");
        }
    }
    printf("\nc=%d",c);
    EndTextPrinting();
    Stop();
}
```

表— 6　受理，非受理判定の結果　(1)

```
a={421122322123}
1110111010100011101111100...    ...-
111000010101...    ...-
01...  ... +
111110100100000101...    ...-
000001100101010011101100...    ...-
000111000101111100...    ...-
101011001101111011000111101...    ...-
10101110110011010011...    ...-
1001111011001111010011010100...    ...-
1000001111111111100110001...    ...-
1101111001000001110...    ...-
011011011110001101100010l...    ...-
11110...    ...-
1111011...    ...-
011011110001101011001111l...    ...-
011111010100110000l...    ...-
1101110010001010010010001010l...    ...-
0010010...    ...-
001101000100000101011100O...    ...-
100111...    ...-
000000101100000010110111O...    ...-
001110001001100l1...    ...-
100001111...    ...-
0010100101101...    ...-
000101110100100111l...    ...-
001001101...    ...-
1011011010100101101011111101...    ...-
010100110...    ...-
010011000101010010010001l1...    ...-
011000111010001111111O...    ...-

c=1
```

表— 7　受理，非受理判定の結果　(2)

```
a={421122322123}
11...    ...-
100...    ...-
1000100100010100101O...    ...-
1010101010010000010111100O0...    ...-
00001110...    ...-
100111100000...    ...-
00...    ...-
00...    ...-
101001100111101001...    ...-
1001110011...    ...-
100001111111110...    ...-
000010000011101101l...    ...-
111010001111100110000111101...    ...-
111111101010000101110...    ...-
111110010010100010010000000l1...    ...-
011011000100100010110010000...    ...-
010010100001010111011...    ...-
111001011111110...    ...-
11101...    ...-
1001110...    ...-
000000011110000101001100...    ...-
10100...    ...-
100001100110110010101O...    ...-
1...    ...-
101111000011010l1...    ...-
00100010...    ...-
10101110011...    ...-
0...    ...-
101111011011100l0...    ...-
000000010110011010000000...    ...-

c=0
```

表— 8　記号列判定プログラム　(2)

```
#include  <SCL.h>
#include  <stdio.h>
#include  <stdlib.h>

main()
{
    int  a[12]={4,2,1,1,2,0,3,3,0,2,1,3},s,d=3,n=2;
    int  w[30][30];
    int  i,j,t,x,l,c=0;
    StartTextOnly();
    BeginTextPrinting();
    printf("a={");
        for(i=0;i<12;i++)
            printf("%d",a[i]);
            printf("}\n");

    for(i=0;i<30;i++)
    {
        s=0;
        srand(TickCount());
        l=(double)rand()/32767*29+1;
        for(j=0;j<l;j++)
        w[i][j]=(int)rand()/16390;
        for(j=0;j<l;j++)
        {
            x=w[i][j];
            t=n*s+x+3;
            s=a[t];
        }

        for(j=0;j<l;j++)
            printf("%d",w[i][j]);
        if(s==d)
        {
            printf("...   ... +\n");
            c++;
        }
        else
        {
            printf("...    ...-\n");
        }
    }
    printf("\nc=%d",c);
    EndTextPrinting();
    Stop();
}
```

る。従ってこれを最後まで繰り返すと$f^*(s_0, w[0]w[1]...w[j])$を得る。

このように染色体表現を使用すると容易に判定可能となる。表—6，7がその結果であるが，これを見ればすぐわかるようにこのオートマトンの受理する記号列は全体の記号列のなかでずいぶんすくない。それを反映してcの値が0に近い。

オートマトンを表—8のように変更すればcの値は上がる。このオートマトンは0，1がそれぞれ奇数個からなる記号列を受理する機械である。状態図で表現すると図—4のようになる。

オートマトンをAとするとその染色体表現$g(A)$は

$g(A)=\{4, 2, 1, 1, 2, 0, 3, 3, 0, 2, 1, 3\}$

である。また，

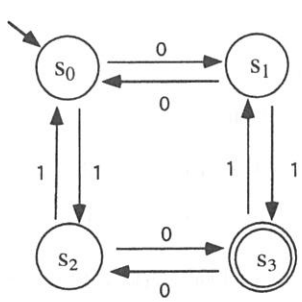
$L(A)=\{w|w\in\{0, 1\}^*, w$のなかに0と1がいずれも奇数個ある。$\}$

である。

表—9では$c=3$で，cの値がすこしあがっていることがわかる。

このプログラムを次のように変更する。染色体表現を与えれば有限オートマトンが確定するから，プログラムの中で染色体表現を任意に作成し，このオートマトンによって，ランダムに作成した記号列の受理，非受理を決定するプログラムを作成する。表—10がその結果である。

表—11は同じ記号列の集合に対して異なるオートマトンの集団を生成して，それらオートマトンの受理能力を比較したものである。基本的なプログラム（表—5）を作成しているのでこのような拡張は容易である。



図—4　オートマトン

表—9　受理，非受理判定の結果　(3)

```
a={421120330213}
1101001...   ...-
11111011011011100011001111000...   ...-
01100011100001100101000001...   ...-
001101110100111111...   ...-
0000111010110101...   ...-
0011111111010...   ...-
01111100...  ... +
11100010100111...   ...-
11100111010...   ...-
11000...   ...-
110000010...   ...-
01100110110000000...   ...-
010010000110100...   ...-
10010010111...   ...-
00000...   ...-
0011011011001000000011010...   ...-
0011000111101...   ...-
00011110...   ...-
011001100010111...   ...-
111100111100...   ...-
11000000111001001110...  ... +
01101110101111011011...   ...-
101000...   ...-
0010101001111...   ...-
10000010010000100091011...  ... +
00010010100101001111011100...   ...-
000000010110010101010000111...   ...-
11...   ...-
1101001000001100000...   ...-
11111011011001110011...   ...-

c=3
```

表—10　受理，非受理判定の結果　(4)

```
a=921<33280584026404840l>5...
0111... ... +
011110... ... -
11110100110011010011001O... ... -
0111... ... +
011011... ... -
111111111... ... -
1111001111111... ... -
111... ... -
0101110010001111111... ... -
11011001110110001001011001111... ... +
001101010... ... -
000110101111100... ... -
0001... ... -
001101... ... -
001101110... ... -
000001010110110011100011000O... ... -
000101000000111101110000O... ... -
001101100101011111100... ... -
0101101111011... ... -
010000001100100000l... ... -
01110101111... ... +
0101001010001011... ... -
110101010010001010110110... ... -
111111111011110111111... ... -
11100010001001... ... -
1110... ... -
000000111110110110011... ... -
000011000110011... ... -
000001001011001101111l1... ... -
1... ... -
 c= 4...c/30=0.133333
```

表—11　オートマトンの受理能力の比較

```
a0=221<1111>1...
a1=621<500015350510>2...
a2=421<33103122>0...
a3=121<00>0...
a4=621<013544515453>3...
a5=321<011112>0...
a6=521<0440303103>2...
a7=621<015513144230>4...
a8=821<1250120652000143>0...
a9=521<1400340334>4...
a10=721<21002240140066>1...
a11=521<1420443311>0...
a12=621<213022114000>1...
a13=321<100222>0...
a14=521<2322302420>1...
a15=621<303123510554>3...
a16=421<23002331>1...
a17=921<6342761745541466648>0...
a18=721<54356432024644>6...
a19=521<4202041133>3...
a20=221<1100>1...
a21=721<65642365502235>1...
a22=521<4104400243>0...
a23=621<541433352212>1...
a24=421<01230022>0...
a25=521<0441114123>1...
a26=721<16531043334126>4...
a27=921<161050537523675176>8...
a28=321<012102>1...
a29=421<10001020>2...
0
10101110100101110
001011100111110111111011
111
01000111101111010100
11100111000000001001100110000 1
1110100011111101
11110111100001011100 1101
0100010110001011000110
0000010011101000110
10100001001000101110110001111 1
0
001111
00100000000110110011011010
01001011110000001010000 00
11101011
11001101001010110001011010 1111
010101101001001
11001011110011010100 1001
00100111111010011010 11
1011
100100110
0000111100 10100
10110011011101001111001 10
11000101111101111010111 11
11100011110110111010
01110110101110000 11
11110001010011110111010010 0000
0111010010100 01
10100001101011110010 1011
```

```
c[0]= 0...c[0]/30=0.000000
c[1]= 0...c[1]/30=0.000000
c[2]= 0...c[2]/30=0.000000
c[3]= 0...c[3]/30=0.000000
c[4]= 0...c[4]/30=0.000000
c[5]= 0...c[5]/30=0.000000
c[6]= 0...c[6]/30=0.000000
c[7]= 30...c[7]/30=1.000000
c[8]= 0...c[8]/30=0.000000
c[9]= 0...c[9]/30=0.000000
c[10]= 30...c[10]/30=1.000000
c[11]= 0...c[11]/30=0.000000
c[12]= 30...c[12]/30=1.000000
c[13]= 30...c[13]/30=1.000000
c[14]= 0...c[14]/30=0.000000
c[15]= 0...c[15]/30=0.000000
c[16]= 0...c[16]/30=0.000000
c[17]= 0...c[17]/30=0.000000
c[18]= 0...c[18]/30=0.000000
c[19]= 30...c[19]/30=1.000000
c[20]= 0...c[20]/30=0.000000
c[21]= 0...c[21]/30=0.000000
c[22]= 0...c[22]/30=0.000000
c[23]= 0...c[23]/30=0.000000
c[24]= 0...c[24]/30=0.000000
c[25]= 0...c[25]/30=0.000000
c[26]= 0...c[26]/30=0.000000
c[27]= 0...c[27]/30=0.000000
c[28]= 0...c[28]/30=0.000000
c[29]= 0...c[29]/30=0.000000
```

5．オートマトンの近似度

2個のオートマトンAとBを与える。またXの上の記号列の集合Wを与えたとき，オートマトンAが受理したWの記号列の個数をt_pとする。Wの濃度 $\#(W)$ から t_p を引いた数をt_mとする。t_p の値が0でないとき，Aが受理した記号列のうちでオートマトンBが受理する記号列の個数をc，さらに t_m が0でないとき，Aが受理しない記号列の中でBが受理しない記号列の個数をdとする。このとき，値

$$e=\frac{c}{t_p}\times\frac{d}{t_m}\times 1000$$

をBの（Wに関する）Aに対する近似度という。$t_p=0$ のときには $e=\frac{d}{t_m}\times 1000$ を，$t_p=\#(W)$（したがって$t_m=0$）のときには$e=\frac{c}{t_p}\times 1000$ を近似度として採用する。

表—12　オートマトンの近似度計算プログラム

```
#include  <SCL.h>
#include  <stdio.h>
#include  <stdlib.h>

main()
{
    int  ta[8]={2,2,1,0,1,0,1,1},tc;
    int  a[5][22],wl[30],wp[30];
    int  m[5],n[5],p[5],s;
    int  w[30][30];
    int  i,j,k,t,x,l,L[5];
    double  tp,tm,c[5],d[5],est[5];
    StartTextOnly();
    BeginTextPrinting();
for(k=0;k<5;k++)
{
    srand(TickCount());
    m[k]=(double)rand()/32767*9+1;
    n[k]=2;
    p[k]=(double)rand()/32767*m[k];
    L[k]=m[k]*n[k]+4;
    a[k][0]=m[k];
    a[k][1]=n[k];
    a[k][2]=1;
    for(j=3;j<=(L[k]-2);j++)
    a[k][j]=(double)rand()/32767*m[k];
    a[k][L[k]-1]=p[k];
    printf("a%d=",k);
        for(j=0;j<=2;j++)
        printf("%d",a[k][j]);
        printf("<");
        for(j=3;j<=L[k]-2;j++)
        printf("%d",a[k][j]);
        printf(">%d\n",a[k][L[k]-1]);
}
for(i=0;i<30;i++)
    {
        srand(TickCount());
        l=(double)rand()/32767*30+1;
        wl[i]=l;
        for(j=0;j<l;j++)
        w[i][j]=(int)rand()/16390;
        for(j=0;j<l;j++)

            printf("%d",w[i][j]);
        printf("\n");
    }
    tc=0;
for(i=0;i<30;i++)
    {
        s=0;
        for(j=0;j<wl[i];j++)
        {
            x=w[i][j];
            t=2*s+x+3;
            s=ta[t];
        }
        if(s==1)
        {
            wp[i]=1;
            tc++;
        }
        else
```

表—12（つづき）

```
        }
        tp=tc;
        tm=30-tp;
        printf("tp=%d\n",tc);

for(k=0;k<5;k++)
{       c[k]=0;
        d[k]=0;
    for(i=0;i<30;i++)
    {
            s=0;
            for(j=0;j<wl[i];j++)
            {
                x=w[i][j];
                t=n[k]*s+x+3;
                s=a[k][t];
            }
            if(wp[i]==1&&  s==p[k])
                c[k]++;
        else
            if(wp[i]==0&&s!=p[k])
                d[k]++;

    }

    if(tp==0)
        {
            est[k]=d[k]/tm*1000;
        }
    else
        { if(tm==0)
            {
                est[k]=c[k]/tp*1000;
            }
            else
            {
                est[k]=(c[k]/tp)*(d[k]/tm)*1000;
            }
        }

    printf("  est[%d]=  %lf\n",k,est[k]);

}
    EndTextPrinting();
    Stop();

}
```

表—12は1つのオートマトン $A(g(A)=2,2,1,0,1,0,1,1)$ に対して5個のオートマトンを無作為に作り30個の記号列に対して受理，非受理の判定をしてAに対する近似度を計算するプログラムである。

表—13，14がその結果である。

著者の1人（矢倉）は近似度を高めるため遺伝的アルゴリズムを採用して良好な結果を得た。その結果は別に報告する。

表—13　オートマトンの近似度計算結果　(1)

```
a0=121<00>0
a1=121<00>0
a2=621<113412501300>0
a3=821<2701527745514517>2
a4=121<00>0
00100111111010100010 1
10100100001000110010110011111
0100000111
11010001111111011100000000010
0100111000111011101110000
1101010100100001000
0101110010011011011111110000
11111010
01100110000
11010011100110011001011110001
10001100001111010011011111
10100010101000100011100
0000010100001111001101100101 01
00101011001010111110100000011
0000000111110111010011001
011100111010011000
11010011001011000101101100
11000001001101001010
1110100011010
111001110001000001
0100001110011010001 0001010
0001000110000010 1001
0001100001001
001111111011010001
0000110111
00100100010001
001001
10111001000110100 1000010
001
1001011101101101100
tp=16
 est[0]= 0.000000
 est[1]= 0.000000
 est[2]= 446.428571
 est[3]= 160.714286
 est[4]= 0.000000
```

表—14　オートマトンの近似度計算結果　(2)

```
a0=821<1420106160360112>6
a1=921<244238142387525653>2
a2=721<23005055111156>1
a3=921<425420755173173872>4
a4=621<355353523433>2
01100011000
11011010100001110110100000000 1
01111100110
1111101110100100
01100111
11101011101
110000001011011
0100001110000010001010
000101101000111
0001011000011100 1011
0011111111110
000110000010001001
0011101010
10000010001110
00001
0010101
10001111110
10110101011010001
11001010
11000110011
1
0110000100110000
0101110101110101111 0001
0100000011111110
1101110001101011011 1101
11011010110100101
1111000010001100000101000
0010
0001100000 1001100000
1011000111001
tp=16
 est[0]= 0.000000
 est[1]= 40.178571
 est[2]= 232.142857
 est[3]= 0.000000
 est[4]= 375.000000
```

参考文献

[1] S. Huzino, On inferring dynamics of automata from finite samples, Mem. Fac. Sci., Kyushu Unv., Ser. A, *XXX*II (1978), 291-299.

[2] 和田健之介，デジタル生命の進化，岩波 (1994)，pp.134。

[3] 高橋智，Macintoshではじめる C，牧野書店 (1995)，pp.209。