制約つき再生可能な文字列分解にもとづく計算機ファイルのデータ圧縮†

森田啓義†† 小林欣吾††

高い圧縮効果と高速性を合わせ持ったデータ圧縮符号として，最近 Fiala-Greene 符号（FG 符号）が注目されている．FG 符号は Ziv-Lempel 符号の一変形であり，すでに符号化ずみの長さ M の過去の文字列の中から，次の入力文字列とできるだけ長く一致する部分（参照文字列）を見つけ出すことによって入力文字列を部分文字列（単語と呼ぶ）の列に分解し，つぎに各単語に対応する参照文字列の存在する位置とその長さを適当な符号で符号化する方式である．本論文では，FG 符号において用いられている文字列分解法を，Lempel と Ziv によって与えられた文字列の再生可能性を一般化した制約つき再生可能性にもとづくものとして位置づけることにより，統計的な性質が未知なデータに対して，そのデータから推定される確率モデルのエントロピに平均圧縮率が収束するという意味で，FG 符号（厳密には FG 符号を簡略化したもの）が有限次数の定常マルコフ情報源のクラスに対して $O(\log\log M/\log M)$ のオーダで漸近最良であることを導く．さらに，FG 符号の実用化をより高めるために，参照文字列の位置の符号化に関して改良を加え，その符号器・復号器のプログラムを実際に組み，いくつかの計算機ファイルに対して圧縮実験を行った結果，従来の圧縮プログラムに比べて 5～20% 高い圧縮効果がえられたことを報告する．

1. はじめに

Ziv と Lempel によって提案されたユニバーサル符号化法[1),2)]は，計算機ファイルの圧縮に効果的であることが広く認められ，その高速性と相まって，理論的な面だけでなく実用の面からも次第に関心が高まり，数多くの変形や，より圧縮効果を高める工夫が提案されている[3)~10)]．なかでも，Fiala-Greene 符号（FG 符号）[8)]は高い圧縮効果と高速性を合わせ持った符号として最近注目されている．

FG 符号は ZL 符号と同様に辞書式符号化法[10)]の一種であり，すでに符号化ずみの過去の文字列の中から，次の入力文字列とできるだけ長く一致する部分（参照文字列）を見つけ出すことによって入力文字列を部分文字列（単語と呼ぶ）の列に分解し，つぎに各単語に対応する参照文字列の存在する位置とその長さを適当な符号で符号化する方式である．さらに，FG 符号には，1）参照文字列はすでに切り出された単語の先頭のみからはじまる，2）参照できる過去の範囲は一定の大きさ M に制限される，という二つの制約が課せられている．これらの制約のため，ZL 符号とは異なり，FG 符号の参照文字列はかならずしも最長一致文字列にはならない．しかし，実用上は，かえって制約があるがゆえに，通常の文字列照合法[11)]や suffix 木[12)]を用いる方法[1),3)]でなく，トライ（Trie）を一般化したパトリシア[12)]を用いて参照文字列をその長さに比例した比較回数で高速に探索することができる．また，圧縮効果に関しても，パトリシアの木構造を利用して参照文字列の長さを効率よく符号化できるため，幅広い種類のデータ，とくに計算機上のテキストファイルやソースファイルに対して高い圧縮効果が期待される[8),10)]．

従来，ZL 符号は定常エルゴード情報源に対して漸近最良であること，すなわち，1 シンボルあたりの平均符号長がエントロピレートに収束することが知られているが，ZL 符号では符号器および復号器のメモリ容量が無制限に大きくなることを許しており，実用的でない．そこで，メモリ容量は一定の値以下に抑えた ZL 符号の変形が数多く提案されてきている[9),10)]．しかしながら，いままで理論的な解析はほとんど行われていない．最近，Wyner と Ziv は，ZL 符号の一変形として，あらかじめ一定長さのデータから作成された文字列の辞書を符号器と復号器で共有し，その辞書に基づいて文字列分解を行う符号（WZ 符号）を提案している[21)]．WZ 符号では辞書は固定されており途中で変更されない点が通常の ZL 符号と異なる．彼らは，WZ 符号がマルコフ情報源を含むある定常エルゴード情報源の 1 クラスに対して漸近最良であることを示した．

† Data Compression of Computer Files Based on Restricted Reproducible Parsing Algorithms by HIROYOSHI MORITA and KINGO KOBAYASHI (Department of Computer Science and Information Mathematics, Faculty of Electro-Communications, The University of Electro-Communications).

†† 電気通信大学電気通信学部情報工学科

本論文では，FG 符号における文字列分解法を，文字列の再生可能性 (reproducibility)[13] を一般化した，制約つき再生可能性にもとづくものとして位置づけることにより，統計的な性質が未知なデータに対して，そのデータから推定される確率モデルのエントロピに平均圧縮率が収束するという意味で，FG 符号が有限次数の定常マルコフ情報源のクラスに対して $O(\log\log M/\log M)$ のオーダで漸近最良であることを導く（なお，本文を通じて，明示しないかぎり対数の底は2とする）．

さらに，FG 符号の実用化をより高めるために，参照文字列の位置の符号化に関して改良を加えた符号を提案する．この改良 FG 符号の符号器・復号器のプログラムを実際に組み，いくつかの計算機ファイルに対して圧縮実験を行った結果，従来の圧縮プログラムに比べて 5〜20% 高い圧縮効果がえられた．

以下，2章では，まず，FG 符号で行われる文字列分解法を，文字列の制約つき再生可能性という概念に基づいて説明し，つぎに，その分解法を効率よく実現するためのデータ構造としてのパトリシアを概説した上で，FG 符号における実際の符号の割り当て方を示す．3章では，FG 符号よりは性能は劣るがより基本的と思われる簡略 FG 符号を定義したうえで，簡略 FG 符号が有限次数のマルコフ情報源に対して漸近最良であることを示す．4章では，実用面での FG 符号の改良を行い，従来の符号との圧縮性能を比較するために行った実験結果を報告する．

2. Fiala-Greene 符号

2.1 文字列分解

K ($<\infty$) 個のシンボル $a_1, a_2, \cdots, a_K$ からなるアルファベット A に対して，A 上の有限長の文字列全体の集合を A^* とおく．A^* は長さ0の空列 λ も含むとする．

$S\in A^*$ は，シンボルの並びとして，$S=s_0s_1\cdots s_{n-1}$, $s_i\in A, 0\leq i<n$ と表される，n を S の長さとよび，$|S|$ と記す．S の部分列，$s_is_{i+1}\cdots s_j$ $(0\leq i\leq j<n)$ を $S[i,j]$ と記す．$S[i,j]$ の先頭文字 s_i の添字 i を $S[i,j]$ の先頭位置と呼ぶ．$0\leq i\leq j<n$ を満たさない i, j に対しては，$S[i,j]=\lambda$ とおく．$0\leq i<n$ に対し，$S[0,i]$ を S の接頭語，$S[i,n-1]$ を S の接尾語という．また，シンボル s_i を先頭とする S の接尾語を，長さを明示せずに $S[i,-]$ と記す場合もある．

$Q, R\in A^*$ に対し，ある $S\in A^*$ が存在して，$S[0,|Q|-1]=Q$ かつ $S[|Q|,|Q|+|R|-1]=R$ のとき，S を Q と R の連結と呼び，QR と記す．さて，集合 $\{0,1,\cdots,n-1\}$ の部分集合 I が与えられているとする．このとき，任意の長さ n の文字列 Q に対して，ある $R\in A^*$ が，

$$\exists i\in I\ [R=(QR)[i,i+|R|-1]] \qquad (1)$$

をみたすとき，R は制約 I のもとで Q から再生可能といい，$Q\xrightarrow{I}QR$ と記す．$Q\xrightarrow{I}QR$ ならば，先頭が Q に含まれる部分列で R と一致するものが少なくとも一つは QR に存在する．それらの先頭位置で最大のものを $q_I(R;Q)$ とおき，R の参照位置と呼ぶ．なお，$I=\{0,1,\cdots,n-1\}$ のときは，ここで与えた制約つき再生可能性の定義は，Ziv と Lempel による文字列の再生可能性に完全に一致する．

制約つき再生可能性を用いて，与えられた入力列 $s_0s_1\cdots s_{n-1}$ をその部分列の列に分解する方法を**図1**に示す．以下では，この方法を算法Pと呼ぶ．また，算法Pによって分解された部分列を単語と呼ぶ．算法P

```
begin
  i := 0; j := 1; V := λ; W := λ; I := φ;
  while i < n do
    begin
      if V ⊥→ VWs_i then
        W := Ws_i
      else
        begin
          I := I ∪ {|V|};
          if W = λ then                                  10
            begin { 直接モード }
              output s_i as W_j; j := j + 1;
              V := Vs_i
            end
          else
            begin { 間接モード }
              output W as W_j; j := j + 1;
              V := VW; W := λ; i := i − 1
            end
        end;
      I := I ∩ {i−M+1,...,i};
      i := i + 1
    end;                                                 20
  if W ≠ λ then
    output W as W_j
end.
```

図1 制約つき再生可能性にもとづく文字列の分解法P
Fig. 1 Parsing algorithm P based on restricted reproducibility.

の9行目においては，新しく切り出された単語の先頭位置 $|V|$ が I に加えられる．21行目では，必要なメモリ量を入力の長さによらず一定にするために，過去の単語の先頭位置の中で，現在の位置 i から M だけ遡った参照可能範囲に含まれるものしか見ることができないという制約を課している．また，17行目において切り出された W_j の直後の s_i から次の単語が始まるように，18行目で i を1減らしている．

[算法Pによる文字列分解の例] $A=\{0,1\}$，$M\geq 8$ の場合，$S=010101011011$ を算法Pで分解すると，単語の列 0, 1, 010101, 101, 1 が得られる．ここで，先頭の s_0 (=0) と s_1 (=1) は参照する文字列がないためにそれぞれが長さ1の単語となる．$S[2,7]=010101$ は $S[0,1]=01$ というパターンを繰り返してえられるが，s_8 が1のために $S[2,7]$ が単語となる．さらに，$S[1,3]$ を参照して $S[8,10]$ が，さらに残りの s_{11} が長さ1の単語として切り出される．

つぎに，$M=4$ の場合を考えると，S は，0, 1, 010101, 1, 0, 1, 1 と分解される．s_7 までの分解は，$M\geq 8$ の場合と同じであるが，s_8 (=1) 以降の文字列の分解が異なる．参照できる範囲は，4時刻前までの文字列 $S[4,7]=0101$ であり，しかも，参照可能な位置は単語の先頭に限る．ところが，$S[4,7]$ に含まれる二つの1はいずれも単語の先頭ではないため，s_8 から参照できる文字列が存在しない．そこで，s_8 は長さ1の単語として切り出される．同じ理由から s_9 (=0) も長さ1の単語として切り出される．なお，s_{10} と s_{11} も長さ1の単語として切り出されるが，s_8 や s_9 の場合と異なり，参照可能範囲内に参照文字列が存在している．すなわち，同じ長さ1の単語でも，切り出される理由には二通りあることに注意する．

2.2 パトリシア

算法Pを効率よく実現するために，FG 符号では，トライ (trie) の概念を一般化したパトリシア (Patricia: Practical algorithm to retrieve information coded in alphanumeric)[14] を利用している．これにより，新しく切り出すべき単語の探索をその長さに比例した比較回数で行える．それゆえ，符号化ならびに復号化処理に要する計算量は入力文字列の長さに比例する．パトリシアとは，つぎの三つの条件を満足する木構造のことである．

(P1) どの中間節点からも2本以上の枝が分岐している．

(P2) 各枝には長さ1以上の文字列（ラベル）が対応づけられている．

(P3) 根または中間節点から分岐する各枝のラベルの先頭文字は相異なる．

パトリシアを用いて算法Pを実行するには，まず，S からすでに分解された各単語 $W_1, W_2, \cdots, W_m$ を先頭とする m 個の S の接尾語を考え，それらの接尾語の中から，その先頭文字が W_m の末尾から数えて M 文字の中に含まれているものだけを取り出す．つぎに，(P1)～(P3)の条件を満たしながら，取り出された接尾語をパトリシア上の根から葉へ向かう枝の列（パスと呼ぶ）に1対1に対応させる操作を新しい単語が切り出されるごとに逐次的に行う．例として，先の文字列 $S=010101011011$，$A=\{0,1\}$，$M=8$ を算法Pで分解した結果のパトリシア表現を**図2**に示す．

図中の太い丸印は根を，普通の丸印は中間節点，四角印は終端節点(葉)を示している．根と中間節点および葉の肩には，それらが生成された順位に通し番号がふられている．ただし，根にはつねに0という番号が割り振られ，中間節点は番号1から始まるものとする．一方，葉は番号0から始まる．以後では，これらの番号と節点そのものとを同一視する．

葉に書き込まれた数字は，根から葉に向かう枝の列(パス）に対応した接尾語の S における先頭位置を示している．この対応関係から，各枝には接尾語の部分列が自然に対応する．例えば，根から左の中間節点1に向かう枝 E_1 に注目する．E_1 を含むパスは2本あり，それぞれには，S 上の位置0から始まる接尾語 $S[0,-]$ (=0101010…) と位置2から始まる接尾語 $S[2,-]$ (=0101011…) が対応している．二つの接尾語の先頭部分の 010101 は共通で，かつ 010101 を先頭にもつ接尾語はこの二つに限るので，E_1 には

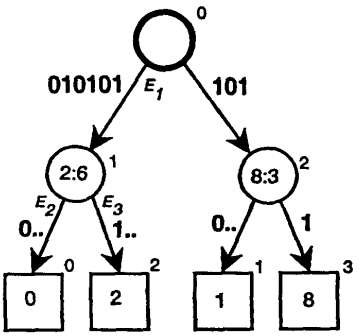

図2 単語の列 0, 1, 010101, 101, 1 のパトリシア表現 ($M\leq 8$ の場合)

Fig. 2 The Patricia representation of the parsed words 0, 1, 010101, 101, 1. $M\leq 8$

```
    T  :  パトリシア
    r  :  T の根から始まるパス
   S_r :  パス r に含まれる枝のラベルを連結した文字列
    L  :  S_r の中で文字列 W の末尾を含むラベル
  v, w :  ラベル L に対応する枝の始点と終点
  p, d :  W の入力列 S 上の先頭位置とラベル L 上における末尾位置 (0 ≤ d < |L|)
   l_v :  根から中間節点 v までのパスに対応する文字列の長さ
begin
 根だけからなる木 T を生成する; i := 0; j := 1; W := λ;
 while i < n do
   begin
     if S_r[0,|W|] = Ws_i を満たすパス r が存在する then
       W := Ws_i
     else
       begin
         if W = λ then
           begin                                                          10
             根から葉 x を分枝し, x に i を書き込む;
             output s_i as W_j; j := j + 1
           end
         else
           begin
             if d < |L| - 1 then
               begin
                 v と w の間に中間節点 y を挿入し, y に p:l_v+d+1 を書き込む;
                 y を w とおく
               end;                                                       20
             w から葉 x を分枝し, x に p を書き込む;
             output W as W_j; j := j + 1; W := λ; i := i - 1
           end;
         i-M 以下の開始位置が書き込まれた葉をすべて T から削除する;
         T が (P1)~(P3) を満たすように, 不要な中間節点を T から削除する
       end;
     i := i + 1
   end;
 if W ≠ λ then output W as W_j
end.                                                                      30
```

図 3 パトリシアを用いた文字列分解法 P′
Fig. 3 A parsing algorithm P′ with Patricia.

010101 というラベルがふられている．さらに，S[0, -] の後半部分 '0…' は枝 E_2 に，S[2, -] の後半部分 '1…' は枝 E_3 にふられている．S[1, -] と S[8, -] の関係も図から明らかである．

中間節点に書き込まれた数字の組 $a:l$ は，a がその節点を通るパスの先頭位置の最大値，l が根からその節点までの部分パスに対応した文字列の長さ l を示している．なお，各枝には実際にラベルがふられているのではなく，そのかわり，M時刻前までの入力文字列を大きさMの環状バッファに格納しておき，ラベルが必要になった時点で，葉および中間節点に与えられた情報を用いて参照される．

制約つき再生可能な意味で最長な単語を入力文字列から切り出すためには，上述のように構成されたパトリシアを用いて，新しい入力文字列とパトリシア上のパスにふられたラベルとの間で最長一致文字列の探索を行う．このとき，探索に要する比較回数は(P3)から最長一致文字列の長さに比例する．パトリシア上の操作としては，探索以外に，a）切り出された単語の登録と，b）先頭が参照可能範囲からはずれた接尾語の削除を行う必要があるが，これも全体として文字列の長さに比例した計算量で行える．詳しい算法を P′ として図 3 に示す．算法 P′ はパトリシアを用いて算法Pを実現したものなので，同じ入力文字列に対する両者の出力は完全に一致することに注意する．算法 P′ によってパトリシアが更新される様子を示した例を付録Aに掲げておく．

2.3 符号割り当て

与えられた文字列Sを符号化するためには，基本的には，算法 P′ によって分解された各単語の先頭位置とその長さに適当な符号を割り当てればよい．FG 符号では，パトリシアの内部構造を利用して，単語の先頭位置を節点の順位に，そして，単語の長さを枝の長さに置きかえることによって，より短い符号割り当てを可能にしている．

FG 符号の構成は，算法 P′ の 9 行目において，1) $W=\lambda$（直接モード）が成り立つ場合と，2) そうでない場合（間接モード）に大きく分かれる．直接モードは，参照可能範囲内に存在しないシンボルが新しく現れる場合に生じる．この場合は，そのシンボルを直接出力する．一方，間接モードでは，最長一致文字列の探索は，必ずある枝 e にふられたラベルの途中もしくは最後において終了する．そこで，e の始点と終点をそれぞれ v，w とおくと，間接モードは 2_1) 終点 w が中間節点の場合（中間節点モード）と，2_2) 葉の場合（葉モード）とにさらに細分して考える．実際の符号の割り当てを以下に示す．ただし，長さ 1 の単語はすべて直接モードとみなし，連続した m 個の直接モードを一括して処理する．

1. 中間節点モード
   0+CBTcode $(w, maxNode)$+CBTcode(d, l_e)
   （ただし，第 3 項は，$l_e=1$ の場合には省略）．
2. 葉モード
   1+SSScode $(d\,;\,1,1,14)$
   +CBTcode $(w-sLeaf, nLeaf)$
3. 直接モード
   1+00+SSScode $(m\,;\,0,1,12)+m$ 個のシンボル列

ここで，+は 2 進系列の連結を意味する．$maxNode$ はいままでにパトリシアに現れた中間節点の最大番号，d は e にふられたラベルと入力系列が一致する長さ，l_e は枝 e にふられたラベルの長さを表す．$sLeaf$ は最小の参照可能範囲位置が書き込まれた葉の順位，$nLeaf$ は葉の総数，さらに，m は連続して現れた直接モードの回数である．

以上のように，e の終点の w と d を符号化すると，復号器側では，符号器で用いたのと同じパトリシアを用意すれば，各節点の情報から単語の位置と長さを正しく復元することができる．

w や d などの非負整数を符号化するには，さまざまな自然数の表現法を用いることが可能であるが[15),16)]，FG 符号では，CBTcode と SSScode を用いている．CBTcode (x, max) は $0\leq x<max$ なる非負整数 x に対する 2 値符号であり，$2^c\leq max<2^{c+1}$ なる c と $z=2^{c+1}-max$ に対して，$x<z$ なら x の c 桁の 2 進数展開，$x\geq z$ なら $x+z$ の $c+1$ 桁の 2 進数展開を x に割り当てる[5),8)]．

一方，SSScode $(x\,;$ start, step, stop$)$ は，$0\leq x<(2^{stop+step}-1)/(2^{step}-1)$ なる非負整数 x に対する 2 値符号である[8),10)]．まず，最初の 2^{start} 個の非負整数には長さ start の 2 進系列を，次の $2^{start+step}$ 個の整数には長さ start+step の 2 進系列を，一般に，$2^{start}(2^{jstep}-1)/(2^{step}-1(\leq x<2^{start}(2^{(j+1)step}-1)/(2^{step}-1)$，$0\leq j\leq t\triangleq$(stop−start)/step なる整数 x には，長さ start+$j$step の 2 進系列を，それぞれ辞書式順序にしたがって割り当てる．つぎに，各 x に対応する 2 進系列の先頭には，語頭条件をみたすように，最初の 2^{start} 個の 2 進系列には 1，次の $2^{start+step}$ 個の 2 進系列には 01，その次は 001 というふうに，自然数の unary 表現をつけ加える．ただし，最後の $2^{start+tstep}$ 個の 2 進系列には $\overbrace{0\cdots0}^{t}$ を先頭につけ加える．ところで，FG 符号の葉モードと直接モードにおける符号語はいずれも 1 で始まるが，直接モードの第 2 項の 00 が SSScode $(0\,;\,1,1,14)$ に等しく，かつ常に $d\geq 1$ なので，両者の区別は正しく行えることに注意する．また，SSScode の stop パラメータの値はここでは文献 8) より大きめにとっている．

復号化に関しては，復元された文字列から符号化で用いたのと同じパトリシアを再構成しながら，参照位置と長さのコードから各単語を復元すればよい．詳細は省略する[8),10)]．

3. 符号化定理

前章で述べた FG 符号の符号化性能の解析を容易に行うため，性能は劣るが FG 符号をより簡単化した符号をあらためて定義する．

[FG 符号の簡約化] 算法 P′ を用いて FG 符号と全く同じ文字列の分離を行うが，符号の割り当て方は，算法 P′ の 9 行目の判定に応じて以下の操作に変更する．

1. 直接モードの場合は，1 ビットの 0 に続いて，入力文字自身を長さ $\lceil\log K\rceil$ の 2 進系列として出力し，連続する直接モードを一括して符号化することは行わない．

2. 間接モードは細分化せずに，常に1ビットの1に続いて，単語Wの参照文字列の先頭位置qとその長さlを符号化する．また長さ1の単語を直接モードと見なすことも行わない．

なお，lの符号化には SSScode $(l;1,1,\infty)$ を用いる．一方，qの符号化については，FG 符号の葉モードの場合と同様に，参照文字列を先頭にもつ入力系列の接尾語に対応した葉vの相対順位 $v-sLeaf$ を CBTcode $(v-sLeaf, nLeaf)$ で符号化する．

さて，任意の長さ n (≥ 1) の文字列 $S\in A^*$ が，算法 P′ によって

$$S=W_1W_2\cdots W_{\nu(S)}$$

と分解された上で，上述の簡略 FG 符号によって符号化されたとする．ここで，$\nu(S)$ はSから切り出された単語の総数である．以後はSを固定して考えるので，$\nu=\nu(S)$ とおく．このとき，1シンボルあたりの平均符号長 $\rho(S)$ は

$$\rho(S)=\frac{1}{n}\Big\{\nu+\sum_{i\in I^{(1)}}\lceil\log K\rceil+\sum_{i\in I^{(2)}}\Big\{|\text{CBTcode}\,(m_i-1,n_i)|+|\text{SSScode}\,(l_i;1,1,\infty)|\Big\}\Big\} \qquad (2)$$

と表される．ここで，$I^{(1)}$ は直接モードに属する単語の番号全体の集合，$I^{(2)}$ は間接モードに属する単語の番号全体の集合である．W_i を切り出す時点における制約 I を I_i とすると，m_i は，W_i に等しい参照文字列の先頭位置 q_i の I_i における順位，n_i は I_i の大きさ，l_i は W_i の長さである．以下では，

$$Q(S)\triangleq\sum_{i\in I^{(2)}}|\text{CBTcode}\,(m_i-1,n_i)| \qquad (3)$$

$$L(S)\triangleq\sum_{i\in I^{(2)}}|\text{SSScode}\,(l_i;1,1,\infty)| \qquad (4)$$

とおく．

$\rho(S)$ を評価するために，(2)の右辺の各項を順番に評価していこう．まず，νを評価するために，つぎの補題を用意する．

補題 1 任意の i $(0\leq i<|S|-M)$ に対して，$S[i, i+M-1]$ の部分列となる単語全体の中には，同一のものが高々K個存在する．

(証明) $1\leq i<\nu(S)$ に対して，$H_i=S[0,p_i-1]$ とおく．各 W_i を切り出すときのIを区別するために，I_i と記す．さらに，各 W_i の先頭文字を w_i と記す．このとき，算法P (もしくは P′) の動作より，W_i は H_i から制約 I_i のもとで再生可能な入力文字列中で最長の文字列である．

さて，ある i $(0\leq i<|S|-M)$ が存在して，ある文字列Tと等しい単語が，$S[i,i+M-1]$ のなかに $K+1$ 個以上含まれていたと仮定する．単語の先頭文字は高々K種類しかないので，ある $a\in A$ に対して，$W_j=W_k=T$，かつ，$w_{j+1}=w_{k+1}=a$ となる j,k $(0\leq j<k<\nu(S))$ が存在する．$j\in I_k$ なので，$H_k\xrightarrow{I_k}H_kW_kw_{k+1}$ が成り立つ．これは，W_k が H_k から制約つき再生可能な最長の文字列であることに矛盾する．よって，題意が示せた． □

つぎに，$S\in A^*$ が長さMの部分列 B_i，$i=1,\cdots,L$ によって，$S=B_1B_2\cdots B_L$ と表されるとしよう．以後，つねに $n=ML$ と仮定する．各 B_i に先頭が含まれる単語の数を ν_i とおく．

定理 1 十分大きなMに対して，

$$\nu\leq\frac{n}{(1-\varepsilon_M)\log_K M}$$

が成り立つ．ここで，$\varepsilon_M=(3+2\log_K\log_K M)/\log_K M$ とおいた．

(証明) 補題1と，文献13) の定理2の証明 (p. 77) から，$\nu_i\leq M/\{(1-\varepsilon_M)\log_K M\}$ が導かれ，$\nu=\sum_{i=1}^{L}\nu_i$ より題意をえる． □

補題 2

$$|I^{(1)}|\leq K\frac{n}{M} \qquad (5)$$

(証明) あるシンボル $a\in A$ が直接モードとして処理されたとする．ウィンドウの大きさがMであることから，続くM個の文字が入力されるまで，aがふたたび直接モードとして処理されることはない．よって，n_a を a が直接モードとして現れる回数とすると，$n_a+(n_a-1)M\leq n$ が成り立つ．これを変形すると，

$$n_a\leq\frac{n+M}{M+1}<\frac{n+M}{M}=\frac{n}{M}+1$$

さらに，n_a が整数であることと，$L=n/M$ より，$n_a\leq n/M$ である．aは任意だから，すべてのシンボルについて総和をとると題意をえる． □

補題 3

$$L(S)\leq 2\nu\log\frac{n+\nu}{\nu} \qquad (6)$$

(証明) 式(4)を変形してゆくと，

$$L(S)\triangleq\sum_{i\in I^{(2)}}|\text{SSScode}\,(l_i;1,1,\infty)|\overset{(a)}{=}\sum_{i\in I^{(2)}}2\lfloor\log(l_i+1)\rfloor\leq\sum_{i=1}^{\nu}2\lfloor\log(l_i+1)\rfloor$$

$$\leq 2\nu \sum_{i=1}^{\nu} \frac{1}{\nu} \log (l_i+1)$$

$$\overset{(b)}{\leq} 2\nu \log\left(\sum_{i=1}^{\nu} \frac{l_i+1}{\nu}\right) = 2\nu \log \frac{n+\nu}{\nu}$$

となる．(a)は任意の非負整数 l は，SSScode$(l; 1, 1, \infty)$ によって，長さ $2\lfloor \log (l+1) \rfloor$ の2進系列に変換されることによる．(b)では，Jensen の不等式を用いた． □

補題 4

$$Q(S) \leq \sum_{i=1}^{L} \nu_i \log \nu_i + 2\nu \qquad (7)$$

(証明) 式(3)を変形してゆくと，

$$Q(S) = \sum_{i \in I^{(3)}} |\mathrm{CBTcode}(m_i, n_i)|$$

$$\overset{(a)}{\leq} \sum_{i \in I^{(3)}} \lceil \log n_i \rceil$$

$$\overset{(b)}{\leq} \sum_{i=1}^{\nu} \lceil \log n_i \rceil \qquad (8)$$

となる．(a)は CBTcode の定義から，(b)は総和をすべての i にわたってとることによって導かれる．

ブロック B_i に先頭が含まれる単語を $W_{i,j}$, $1 \leq i \leq L$, $1 \leq j \leq \nu_i$ とおき，$W_{i,j}$ が切り出されるときに参照可能な文字列の数を $n_{i,j}$ とすると，式(8)は，さらに次のように変形される．

$$Q(S) \leq \sum_{i=1}^{L} \sum_{j=1}^{\nu_i} \lceil \log n_{i,j} \rceil$$

$$\leq \sum_{i,j} \{\log n_{i,j} + 1\}$$

$$= \sum_{i,j} \log n_{i,j} + \nu$$

$$\overset{(c)}{\leq} \sum_{i,j} \log (\nu_{i-1} + \nu_i) + \nu$$

$$= \sum_{i} \log (\nu_{i-1} + \nu_i) + \nu$$

$$= \nu \sum_{i} \frac{\nu_i}{\nu} \log\left(\frac{\nu_{i-1}+\nu_i}{2\nu - \nu_L}\right) + \sum_{i} \log (2\nu - \nu_L) + \nu$$

$$\overset{(d)}{\leq} \nu \sum_{i} \frac{\nu_i}{\nu} \log \frac{\nu_i}{\nu} + \nu \log \nu + 2\nu$$

$$= \sum_{i=1}^{L} \nu_i \log \nu_i + 2\nu$$

(c)では，関係式 $n_{i,j} \leq \nu_{i-1} + \nu_i$ $(1 \leq i \leq L, 1 \leq j \leq \nu_i)$ を用いた．ただし，$\nu_0 = 0$ である．(d)は，二つの確率分布 $\{p_i\}$, $\{q_i\}$ について成り立つ不等式，$\sum_i p_i \log q_i \leq \sum_i p_i \log p_i$ を用いた． □

さらに，Sから構成される k 次のマルコフ情報源モデルの経験エントロピと $\rho(S)$ との関係を示そう．まず，k 次マルコフ情報源モデルの状態を $\sigma_i \triangleq s_{i-k} \cdots s_{i-1}$ $(0 \leq i < n)$ とおく．なお，初期状態として，$\sigma_0 = s_{-k} s_{-k+1} \cdots s_{-1} \in A^k$ はあらかじめ与えているものとする．関数 $\delta: A \times A \times A^k \times A^k \to \{0, 1\}$ を，

$$\delta(s, t, \sigma, \tau) \triangleq \begin{cases} 1 & s = t \quad \text{かつ} \quad \sigma = \tau \\ 0 & \text{その他のとき} \end{cases}$$

として定義する．さらに，

$$q_S^k(s, \sigma) = \frac{1}{n} \sum_{i=0}^{n-1} \delta(s_i, s, \sigma_i, \sigma)$$

$$q_S^k(\sigma) = \sum_{s \in A} q_S^k(s, \sigma)$$

$$q_S^k(s|\sigma) = \begin{cases} q_S^k(s, \sigma)/q_S^k(\sigma) & q_S^k(\sigma) > 0, \\ 0 & q_S^k(\sigma) = 0 \end{cases}$$

とおく．Sから定まる k 次の経験エントロピを

$$H(q_S^k) \triangleq -\sum_{\sigma \in A^k} q_S^k(\sigma) \sum_{s \in A} q_S^k(s|\sigma) \log q_S^k(s|\sigma)$$

と定義する．

定理 2

$$nH(q_S^k) \geq \sum_{i=1}^{L} \left\{ \nu_i \log \nu_i - \nu_i \Delta - \nu_i \log \frac{M}{\nu_i} \right\} \qquad (9)$$

ここで，$\Delta = \{1 - \log \ln 2 + (k+1) \log (K+1)\}$ である．

定理2の証明は付録Bで行う．さて，式(7)に(9)を代入して $Q(S)$ の評価を続けると，

$$Q(S) \leq nH(q_S^k) + \sum_{i=1}^{L} \nu_i \log \frac{M}{\nu_i} + (2+\Delta)\nu$$

$$= nH(q_S^k) + \nu \sum_{i=1}^{L} \frac{\nu_i}{\nu} \log \frac{M}{\nu_i} + (2+\Delta)\nu$$

$$\leq nH(q_S^k) + \nu \log\left(\sum_{i=1}^{L} \frac{M}{\nu}\right) + (2+\Delta)\nu$$

$$\leq nH(q_S^k) + \nu \log \frac{n}{\nu} + (2+\Delta)\nu \qquad (10)$$

となる．式(5)，(6)，(10)を(2)に代入すると，

$$n\rho(S) \leq nH(q_S^k) + 2\nu \log \frac{n+\nu}{\nu} + \nu \log \frac{n}{\nu}$$

$$+ \frac{nK}{M} \{\log K + 1\} + (3+\Delta)\nu \qquad (11)$$

をえる．ところで ν は定理1より，Mを十分大きくとると，

$$\nu \leq \frac{Cn}{\log M} \qquad (12)$$

で上から抑えられる．ここで，Cはある正の定数とする．(11)の右辺の第2項と第3項を ν の関数と見なして，

$$f(\nu) = 2\nu \log \frac{n+\nu}{\nu} + \nu \log \frac{n}{\nu} \qquad (13)$$

とおく．簡単な計算から，x^* を $2\ln(x+1)-3\ln x-2(1+x)^{-1}=1$ の解とすると，$f(\nu)$ は $0<\nu<nx^*$ において単調増加であることが分かる．ここで，x^* はただ一つ存在し，$x^*\approx 0.681634$ である．さて，(12)と $C/\log M<x^*$ を同時に満足するように M を十分大きくとれば，(12)の範囲において $f(\nu)$ は単調増加なので，

$$f(\nu)\leq f(Cn/\log M)\overset{(a)}{=}O(n\log\log M/\log M)$$

が成り立つ．ここで，(a)は(13)の右辺に $\nu=Cn/\log M$ を代入して整理することから得られる．さらに，(11)の第4項と第5項も明らかに $O(n\log\log M/\log M)$ である．結局，つぎの符号化定理が証明された．

定理 3 任意の $S\in A^n$ を簡略 FG 符号で符号化した場合，平均符号長 $\rho(S)$ は，Sから求められる k 次マルコフ情報源モデル ($k\leq 0$) の経験エントロピに対して，次の関係をみたす．

$$\rho(S)\leq H(q_S^k)+O\left(\frac{\log\log M}{\log M}\right)$$

定理3からただちに次の系をえる．

系 1 $X=X_0X_1\cdots X_{n-1}$ を k 次マルコフ情報源から出力された確率変数の列とする．このとき，

$$E\rho(X)\leq H(X_k|X_0,\cdots,X_{k-1})+O\left(\frac{\log\log M}{\log M}\right)$$

が成り立つ．ここで，$E\rho(X)$ は $\rho(X)$ の期待値，$H(X_k|X_0,\cdots,X_{k-1})$ は k 次マルコフ情報源の条件つきエントロピである．

4. 符号の実用化

4.1 FG 符号の改良

パトリシアの更新において節点を削除する場合，葉は生成順位にしたがって削除され，残った葉の順位は連続している．それゆえ，FG 符号で行っているように，葉の順位を先頭順位からの相対番地として表せば，各葉に0から $nLeaf-1$ までの番号を対応させることができる．一方，中間節点は生成順に削除されるとは限らないので，FG 符号では，中間節点の順位を符号化するのに高々長さ「log $maxNode$」の2進系列を用いている．パトリシアに含まれる中間節点の数を $nNode$ とすると，$maxNode$ は単調に増加するのに対し，不要になった中間節点の削除があるため，$nNode$ の数はつねに増加するとは限らない．一般には，入力系列長が大きくなれば，$nNode$ は $maxNode$ に比べて十分小さくなることが期待される．

そこで，二つの配列 N2A[0..$M-1$] と A2N[0..$M-1$] を用意し，符号化・復号化算法におけるパトリシアの更新（挿入・削除）の際に以下に示す操作を加える．

1. 中間節点 m を挿入する場合

A2N[$nNode$]:=w;
N2A[w]:=$nNode$;
$nNode$:=$nNode+1$;

2. 中間節点 w を削除する場合

N2A[A2N[$nNode-1$]]:=N2A[w];
A2N[N2A[w]]:=A2N[$nNode-1$];
$nNode$:=$nNode-1$;

ここで，N2A[w] には中間節点 w の変換後の番号 $i(w)$ が，A2N[i] には i ($0\leq i<nNode$) に対応する変換前の中間節点 w が記録される．この操作によって，中間節点の番号は，0から $nNode-1$ までの数 $i(w)$ に変換され，$i(w)$ は高々長さ「log $nNode$」の2進系列によって表現できる．そこで，中間節点モードの符号割り当てを，

0+CBTcode (N2A[w], $nNode$)
+CBTcode (d, l_t)

に変更する．復号時にも，符号化の場合と全く同じ方法で，配列 N2A と A2N を更新しながら，受けとった符号語から i を読み出し，A2N[i] から節点 w を復元すればよい．

4.2 符号化結果

計算機上のいくつかのファイルに対して符号化・復号化の計算機シミュレーションを行った．対象としたデータは，日本語文書ファイル (Jtextdata1～2) とC言語のソースファイル (csrcdata1～2)，英文の文書ファイル (textdata1～3)，そしてあるカラー濃淡画像のR成分 (imagedata1) とB成分 (imagedata2) と，それらを全部まとめてつくったファイル (Alldata) である．これらのデータを，従来の圧縮法と，FG 符号，改良 FG 符号によってバイト単位 ($K=256$) で符号化し，えられた符号の長さを入力データ長さで割った圧縮率をもとめた．従来の圧縮法としては，とくに，UNIX 上で標準的な圧縮ツールとして広く用いられている compress[17] と，フリーウェアとして配布されている LHarc[18] を比較対象とした．compress は LZW 符号[4] を計算機上に実現したものであり，一方，LHarc は移動辞書型の LZ 符号の出力をさらに動的ハフマン符号[19] によって符号化するものである[22]．結果を**表1**に示す．なお，FG 符号，改良 FG

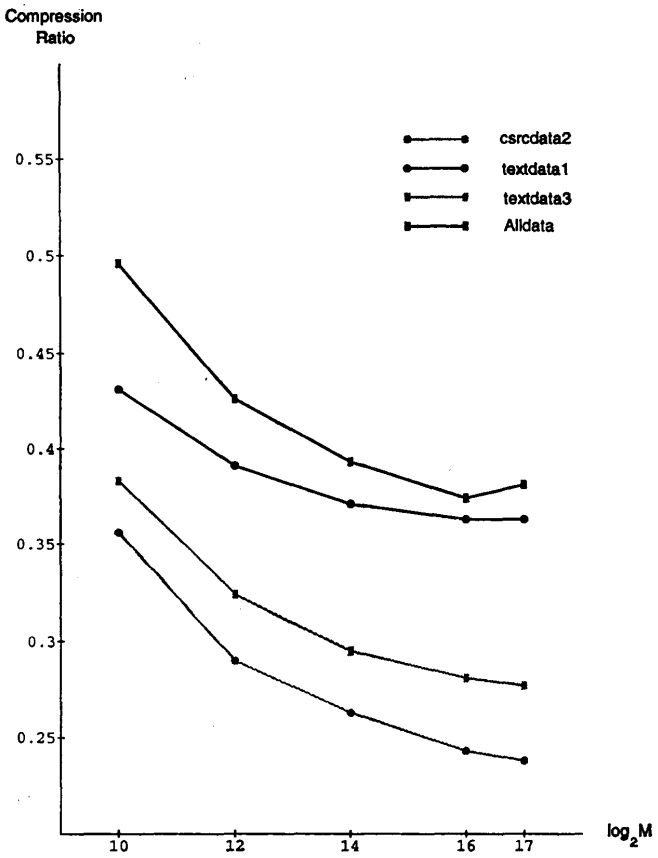

図 4 参照可能範囲 M と改良 FG 符号の圧縮率との関係
Fig. 4 The relationship between window length M and compression ratio of the revised FG codes.

表 1 各種符号に対する圧縮率の比較
Table 1 Comparison of compression ratio for various coding schemes.

$K=256$

ファイル名	サイズ	Compress	LHarc	FG	改良FG
Jtextdata1	10323	.587	.464	.452	.452
Jtextdata2	53271	.503	.416	.367	.367
csrcdata1	29776	.404	.274	.251	.251
csrcdata2	104280	.412	.268	.245	.243
imagedata1	65536	.715	.703	.693	.693
imagedata2	65536	.770	.741	.744	.744
textdata1	46033	.469	.364	.363	.363
textdata2	63997	.467	.420	.203	.203
textdata3	280267	.360	.305	.292	.281
Alldata	719019	.504	.410	.405	.374

符号の参照可能範囲 Mは 2^{16} とした.

改良 FG 方式による改善効果は，ファイルが 100 K バイト以下ではほとんど見られないが，データが大きくなるにつれて次第に現れてくる．約 700 K バイトの Alldata では，8% ほど圧縮率が下がっている．さらに，改良 FG 符号は，compress と比べて 20% 程度，LHarc と比べて 5% 程度，圧縮率が向上している．また，いくつかのデータについて，参照可能範囲Mを変化した場合における改良 FG 符号の圧縮率の関係を図 4 に示す.

また，改良 FG 符号の処理時間は compress のおよそ 5 倍から 10 倍であった．改良 FG 符号のプログラムはすべてC言語によって書かれているので，少なくともポインタ操作の部分をアセンブラで記述すれば，両者の隔たりはさらに縮まると思われる.

5. おわりに

高速性と高い圧縮効果を合わせもつことで知られる FG 符号の符号化性能の評価を行った．FG 符号において用いられている文字列分解法を，文字列の再生可能性を一般化した，制約つき再生可能性にもとづくものとして位置づけることにより，統計的な性質が未知なデータに対して，そのデータから推定される確率モデルのエントロピに平均圧縮率が収束するという意味で，FG 符号（厳密には簡略 FG 符号）が有限次数の定常マルコフ情報源のクラスに対して $O(\log\log M/\log M)$ のオーダで漸近最良であることを導いた．ここで，Mは FG 符号において参照できる過去の範囲である.

さらに，FG 符号の実用化をより高めるために，参照文字列の位置の符号化に関して改良を加え，その符号器・復号器のプログラムを実際に組み，いくつかの計算機ファイルに対して圧縮実験を行った結果，改良 FG 方式は文書ファイルやソースファイルの圧縮に非常に適しており，現在ワークステーション等で広く用いられている他の圧縮法に比べても，5% から 20% ほど圧縮性能が向上することが分かった.

参考文献

1) Ziv, J. and Lempel, A.: A Universal Algorithm for Sequential Data Compression, *IEEE Trans. Inf. Theory*, Vol. 23, No. 3, pp. 337-343 (1977).
2) Ziv, J. and Lempel, A.: Compression of Individual Sequences via Variable-Rate Coding, *IEEE Trans. Inf. Theory*, Vol. 24, No. 5,

pp. 530-536 (1978).
3) Rodeh, M. and Pratt, M.: Linear Algorithm for Data Compression via String Matching, *J. ACM*, Vol. 28, No. 1, pp. 16-24 (1981).
4) Welch, T. A.: A Technique for High-Performance Data Compression, *IEEE Comput.*, pp. 8-19 (June 1984).
5) 横尾英俊：ユニバーサル情報符号化のための修正 Ziv-Lempel 符号，電子通信学会論文誌 (A), Vol. J 68-A, No. 7, pp. 664-671 (1985).
6) Bell, T. C.: Better OPM/L Text Compression, *IEEE Trans. Communications*, Vol. 34, No. 12, pp. 1176-1182 (1986).
7) 朴 志煥，今井秀樹，山森丈範，伊藤秀一，石井正博：パターンマッチングと算術符号を用いた実用的なデータ圧縮法，電子情報通信学会論文誌 (A), Vol. J 71-A, No. 8, pp. 1615-1623 (1988).
8) Fiala, E. and Greene, D.: Data Compression with Finite Windows, *Comm. ACM*, Vol. 32, No. 4, pp. 490-505 (1989).
9) 山本博資：ユニバーサルデータ圧縮アルゴリズムについて，信学会技報告，No. IT 90-97, pp. 7-14 (Jan. 1991).
10) Bell, T. C., Cleary, J. G. and Witten, I. H.: *Text Compression*, chapter 8, Prentice Hall, Inc. (1990).
11) Knuth, D. E., Morris, J. H. and Pratt, V. R.: Fast Pattern Matching in Strings, *SIAM J. Comput.*, Vol. 6, No. 2, pp. 323-350 (1977).
12) McCreight, E.: A Space-Economical Suffix Tree Construction Algorithm, *J. ACM*, Vol. 23, No. 2, pp. 262-272 (1976).
13) Lempel, A. and Ziv, J.: On the Complexity of Finite Sequences, *IEEE Trans. Inf. Theory*, Vol. 22, No. 1, pp. 75-81 (1976).
14) Morrison, D.: PATRICIA—Practical Algorithm to Retrieve Information Coded in Alphanumeric, *J. ACM*, Vol. 15, No. 4, pp. 514-534 (1968).
15) Elias, P.: Universal Codeword Sets and Representations of the Integers, *IEEE Trans. Inf. Theory*, Vol. 21, No. 2, pp. 194-203 (1975).
16) 雨宮 孝，山本博資：新しい自然数のユニバーサル表現のクラスについて，第 13 回情報理論とその応用学会シンポジウム予稿集，pp. 491-496 (1991).
17) Thomas, S. W., McKie, J., Davies, S., Turkowski, K., Woods, J. A. and Orost, J. W.: *Compress (version 4.0) Program and Documentation*, available from joe@petsd. UUCP(1985).
18) 田川洋一：LHarc UNIX V 1.00 ドキュメント (Sep. 1989).
19) Vitter, J. S.: Design and Analysis of Dynamic Huffman Codes, *J. ACM*, Vol. 34, No. 4, pp. 825-845 (1987).
20) Merhav, N., Gutman, M. and Ziv, J.: On the Estimation of the Order of a Markov Chain and Universal Data Compression, *IEEE Trans. Inf. Theory*, Vol. 35, No. 5, pp. 1014-1019 (1989).
21) Wyner, A. D. and Ziv, J.: Fixed Data Base Version of the Lempel-Ziv Data Compression Algorithm, *IEEE Trans. Inf. Theory*, Vol. 37, No. 3, pp. 878-880 (1991).
22) 奥村晴彦，吉崎栄泰：圧縮アルゴリズム入門，C マガジン，Vol. 3, No. 1, pp. 44-68 (1991).

付 録 A

(算法 P′ の実行例)

S=010101011011, M=8 の場合を考える．根だけからなる初期状態 (**図 5** の(a)) から始まって，s_0 が入力されると，算法 P′ の 9 行目において $W=\lambda$ が成り立つので，11, 12 行目が実行され，パトリシアは図 5 の(b)に示すように更新される．つぎの入力 s_1 に対しても，参照文字列は見つからないので，同じく，11, 12 行目が実行され，図 5 の(c)に示すパトリシアがえられる．

さて，根から 0 番目の葉に向かうパス r にふられたラベル $S_r=S[0,-]$ に対して，等式 $S_r[0,k]=s_2\cdots s_{2+k}$, $0\le k\le 5$ が成り立つので，3 行目以下の while 文の中で，i が 2 から 7 までにおいては，5 行目の判定文は真となる．それゆえ，27 行目で $i=8$ となった

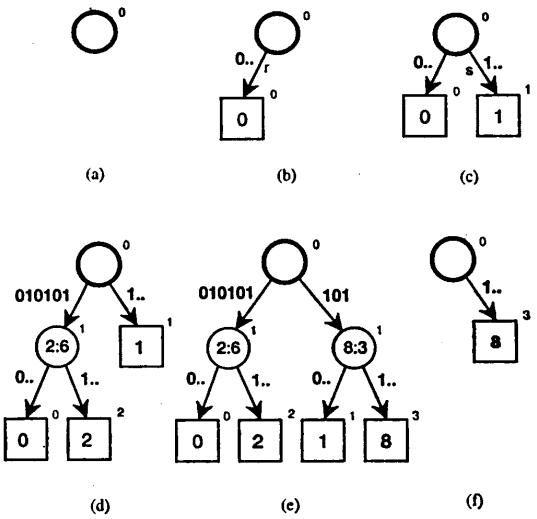

図 5 '010101011011' に対して算法 P′ を実行した場合のパトリシアの遷移
Fig. 5 The transition of Patricia when algorithm P′ runs for the input '010101011011'.

直後では $W=010101$ がえられる．while 文の次の繰り返しでは，$S_r[0,7]\neq Ws_8$ なので9行目以下の else 節が実行され，さらに $W\neq\lambda$ より 16 行目以降が実行される．このとき，d の値は定義より $d=5$，また，$L=12$ なので，16 行目の判定文は真となり，18，19 行目が実行される．その結果，根と葉0の間に新しい中間点0が挿入され，2：6 が書き込まれる（図5の(d)を参照）．ここで，今の場合は，18 行目において p は W の先頭位置なので2，l_v は v が根だから0であることに注意する．さらに，21 行目において，中間節点0から葉2が分岐し，2が書き込まれる．

つぎに5行目の判定文を実行する時点では，$i=8$，$W=\lambda$ である．根から葉1に向かうパス s に対して，$S_s=S[1,-]$ であり，$S_s[0,2]=S[8,10]$ かつ $S_s[0,3]=S[8,11]$ なので，根と葉1の間に中間節点2が挿入され，8：3 が書き込まれる（図5の(e)を参照）．さらに中間節点2から葉3が分岐し，8が書き込まれる．その直後の 22 行目で i は 10 に戻される．24 行目では2以下の開始位置が書き込まれた葉がパトリシアから消去され，さらに 25 行目において，不要な中間節点が削除された結果，最終的にパトリシアは，図5の(f)となる．

付 録 B

(定理2の証明)

文献 20) で引用されている Plotnik と Ziv による証明法にしたがう．いま，A^* 上の確率測度 P として，とくに，$P^k(s|\sigma)=q_S^k(s|\sigma)$，$s\in A,\sigma\in A^k$ なる条件つき確率から定まるものを考える．各 B_i の開始状態を $\sigma^{(i)}\triangleq s_{M(i-1)-M}\cdots s_{M(i-1)-1}(\sigma^{(0)}=\sigma_0)$ とおく．このとき，S の生起確率 $P^k(S|\sigma_0)$ は，$\prod_{i=1}^{L}P^k(B_i|\sigma^{(i)})$ に等しく，さらに，

$$H(q_S^k)=-\frac{1}{n}\log P^k(s|\sigma_0)$$

なので，

$$nH(q_S^k)=-\sum_{i=1}^{L}\log P^k(B_i|\sigma^{(i)}) \tag{14}$$

が成り立つ．以下では，(14)の右辺の各項 $-\log P^k(B_i|\sigma^{(i)})$ を評価しよう．算法Pによって分解された単語 W_k，$k=1,\cdots,\nu$ のうち，B_i に先頭が含まれるものを $W_{i,j}$，$j=1,\cdots,\nu_i$ と記す．ここで，ν_i は B_i に先頭が含まれる単語の数である．ここで，W_{i,ν_i} と B_i とが重なる部分をあらたに W_{i,ν_i} とおく．すると，ある $V\in A^*$ が存在して，$B_i=VW_{i,1}W_{i,2}\cdots W_{i,\nu_i}$ と表される（Vはつねに空系列 λ であるとは限らないことに注意する）．したがって，

$$-\log P^k(B_i|\sigma^{(i)})\geq-\log\prod_{j=1}^{\nu_i}P(W_{i,j}|\sigma_{i,j}) \tag{15}$$

となる．ここで，各 B_i，$i=1,\cdots,L$ に対して，

$W_{i;j}^{l,\sigma}$ ：状態 σ から始まる長さ l の単語のうち最初から数えて j 番目のもの

$\nu_i(l,\sigma)$：長さ l の単語のうち状態 σ から始まるものの数

$l_{max}^{(i)}$ ：最長の単語の長さ

とおくと，(15)の最右辺は，つぎのように表される．

$$\begin{aligned}&-\log\prod_{j=1}^{\nu_i}P(W_{i,j}|\sigma_{i,j})\\&=-\sum_{l=1}^{l_{max}^{(i)}}\sum_{\sigma\in A^k}\sum_{j=1}^{\nu_i(l,\sigma)}\log P(W_{i;j}^{l,\sigma}|\sigma)\end{aligned} \tag{16}$$

ここで，(16)の最も内側の総和を変形してゆくと，

$$\begin{aligned}&-\sum_{j=1}^{\nu_i(l,\sigma)}\log P(W_{i;j}^{l,\sigma}|\sigma)\\&=-\nu_i(l,\sigma)\sum_j\frac{1}{\nu_i(l,\sigma)}\log P(W_{i;j}^{l,\sigma}|\sigma)\\&\overset{(a)}{\geq}-\nu_i(l,\sigma)\log\left(\sum_j\frac{1}{\nu_i(l,\sigma)}P(W_{i;j}^{l,\sigma}|\sigma)\right)\\&\overset{(b)}{\geq}\nu_i(l,\sigma)\log\frac{\nu_i(l,\sigma)}{K+1}\end{aligned} \tag{17}$$

となる．ここで，(a)では Jensen の不等式を，(b)では，$\sum_j P(W_{i;j}^{l,\sigma}|\sigma)\leq(K+1)$ を用いた．この不等式は，補題1と B_i 内の右端の単語 W_{i,ν_i} が切り出しの途中で途切れている場合があることから導かれる．

式(17)を(16)に代入して，(15)をさらに変形してゆくと，

$$\begin{aligned}&-\log P^k(B_i|\sigma^{(i)})\\&\geq\sum_l\sum_\sigma\nu_i(l,\sigma)\{\log\nu_i(l,\sigma)-\log(K+1)\}\\&\overset{(c)}{=}\sum_l\nu_i(l)\sum_\sigma\frac{\nu_i(l,\sigma)}{\nu_i(l)}\left\{\log\frac{\nu_i(l,\sigma)}{\nu_i(l)}+\log\nu_i(l)\right.\\&\qquad\left.-\log(K+1)\right\}\\&\overset{(d)}{\geq}\sum_l\nu_i(l)\log\nu_i(l)-(k+1)\nu_i\log(K+1)\end{aligned} \tag{18}$$

となる．(c)では，$\nu_i(l)\triangleq\sum_\sigma\nu_i(l,\sigma)$ を分子，分母にかけ，(d)では，$\nu_i=\sum_l\nu_i(l)$ と

$$-\sum_\sigma\frac{\nu_i(l,\sigma)}{\nu_i(l)}\log\frac{\nu_i(l,\sigma)}{\nu_i(l)}\leq k\log K\leq k\log(K+1) \tag{19}$$

を用いた．

$\bar{l}_i=\sum_l l\nu_i(l)/\nu_i$ とおいて，式(18)の右辺第1項を変形してゆくと，

$$\begin{aligned}
&\sum_{l=1}^{l_{max}^{(i)}}\nu_i(l)\log\nu_i(l)\\
&=-\nu_i\sum_l\frac{\nu_i(l)}{\nu_i}\log\frac{1}{\nu_i(l)}\\
&\overset{(e)}{=}-\nu_i\sum_l\frac{\nu_i(l)}{\nu_i}\log\frac{2^{-l/l_i}}{\nu_i(l)}-\sum_l\nu_i(l)\log 2^{l/l_i}\\
&=-\nu_i\sum_l\frac{\nu_i(l)}{\nu_i}\log\frac{2^{-l/l_i}}{\nu_i}-\frac{1}{\bar{l}_i}\sum_l l\nu_i(l)\\
&\overset{(f)}{=}-\nu_i\sum_l\frac{\nu_i(l)}{\nu_i}\log\frac{2^{-l/l_i}}{\nu_i(l)}-\nu_i\\
&\overset{(g)}{\geq}-\nu_i\log\left\{\sum_l\frac{1}{\nu_i}2^{-l/l_i}\right\}-\nu_i\\
&=-\nu_i+\nu_i\log\nu_i-\nu_i\log\sum_l 2^{-l/l_i}\\
&\overset{(h)}{\geq}-\nu_i+\nu_i\log\nu_i-\nu_i\log\frac{2^{-1/l_i}}{1-2^{-1/l_i}}\\
&=-\nu_i+\nu_i\log\nu_i+\nu_i\log(2^{1/l_i}-1)\\
&\overset{(i)}{\geq}-\nu_i+\nu_i\log\nu_i+\nu_i\log\left(\frac{\ln 2}{\bar{l}_i}\right)\\
&\overset{(j)}{=}\nu_i\log\nu_i-\nu_i(1-\log\ln 2)-\nu_i\log\frac{M}{\nu_i}
\end{aligned}\tag{20}$$

となる．ここで，(e)は，直前の等式の log のなかに分子，分母とも $2^{l/l_i}$ をかけて整理するとえられる．(f)は $\nu_i\bar{l}_i=\sum_{l=1}^{l} l\nu_i(l)$ より，(g)は Jensen の不等式から導かれる．また，(h)では不等式，

$$\sum_{l=1}^{l_{max}^{(i)}}2^{-l/l_i}=2^{-1/l_i}\frac{1-2^{-l_{max}^{(i)}/l_i}}{1-2^{-1/l_i}}<\frac{2^{-1/l_i}}{1-2^{-1/l_i}}$$

を用いた．(i)では，不等式 $x-1\geq\ln x$ に，$x=2^{1/l_i}$ を代入してえられる不等式 $2^{1/l_i}-1\geq(\ln 2)/\bar{l}_i$ を用いた．(j)では，$\bar{l}_i=M/\nu_i$ なる関係式を用いた．式(20)を(18)に代入すると(9)をえる． □

(平成3年7月26日受付)
(平成3年12月9日採録)

森田 啓義（正会員）

昭和49年大阪大学基礎工学部生物卒業．昭和58年同大大学院博士課程修了．同年豊橋技術科学大学助手．平成2年10月より電気通信大学情報工学科講師，現在に至る．工学博士．3次元物体の計測・認識，画像情報の伝送・加工に関する研究に従事．現在は，情報理論・パターン認識の研究に関心をもっている．電子情報通信学会，情報理論とその応用学会，IEEE 各会員．

小林 欣吾

昭和42年3月東京大学工学部計数工学科卒業．昭和45年3月東京大学大学院工学系研究科修士課程卒業．昭和45年4月より大阪大学基礎工学部生物工学科勤務．平成元年10月より電気通信大学情報工学科助教授．情報理論，グラフ理論，組み合せ理論などの理論的研究とそれらの応用に関する研究に携わる．特に多端子情報理論や有限のパラメータで記述される情報システムの研究に関心を持っている．