# 符号理論･暗号理論

## - No.4 算術符号化 -

渡辺　裕

# Coding Theory / Cryptography

## - No.4 Arithmetic Coding -

Hiroshi Watanabe

# 符号の種類

- 非ブロック符号……（算術符号）
- ブロック符号
  - 特異な符号
  - 非特異な符号
    - 一意復号不可能な符号
    - 一意復号可能な符号
      - 瞬時復号不可能な符号
      - 瞬時復号可能な符号……（ハフマン符号）

# Type of Codes

- Non-block Code..... (Arithmetic Code)
- Block Code
  - Singular Code
  - Non-singular Code
    - Uniquely undecodable Code
    - Uniquely Decodable Code
      - Non-instant decodable code
      - Instant decodable code..... (Huffman Code)

# 算術符号

- 2元情報源のシンボル(“0”,“1”) の系列を任意の長さに区切る
- 先頭に小数点を仮定し、コードワード(符号語)を2進数に見立てる
- 2進数を区間 [0,1] に射影
- 射影区間を一意に決定するための、区間決定ビット数が決まる
- 区間決定コードワードを出力する

# Arithmetic Coding

- Bound sequence of binary source symbol (“0”,“1”) in arbitrary length
- Set fractional point at the head, regard codeword as a binary number
- Project binary number to section [0,1]
- Bit number is counted to determine the section
- Output codeword to specify the section

# 算術符号(2)

- Eliasの符号
  - 算術符号化の原理
  - 実用化には問題点
    - 演算精度の不足
    - リアルタイム復号が不可能
- LR(Langdon-Rissanen)型算術符号
  - 実用化の出発点
    - 演算精度の対策
    - リアルタイム復号可能

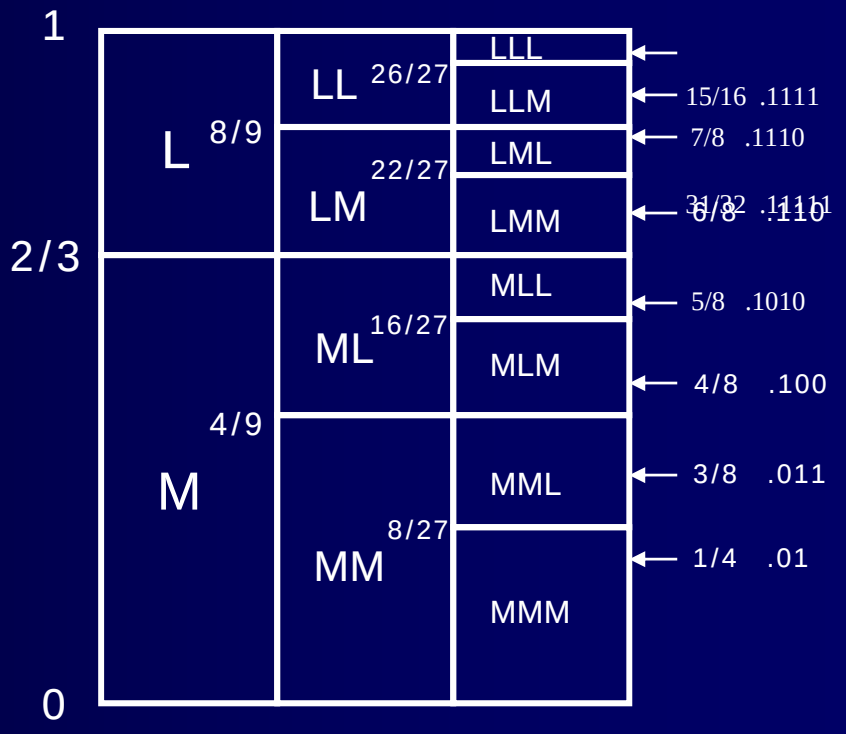
# Arithmetic Coding(2)

- Elias Code
  - Principle of Arithmetic Coding
  - There exists problem for practical use
    - Insufficient operation accuracy
    - Unable to decode in realtime

- LR (Langdon-Rissanen) Type Arithmetic Coding
  - Starting point for practical application
    - Remedy for operational accuracy
    - Decodable in realtime

# Eliasの符号

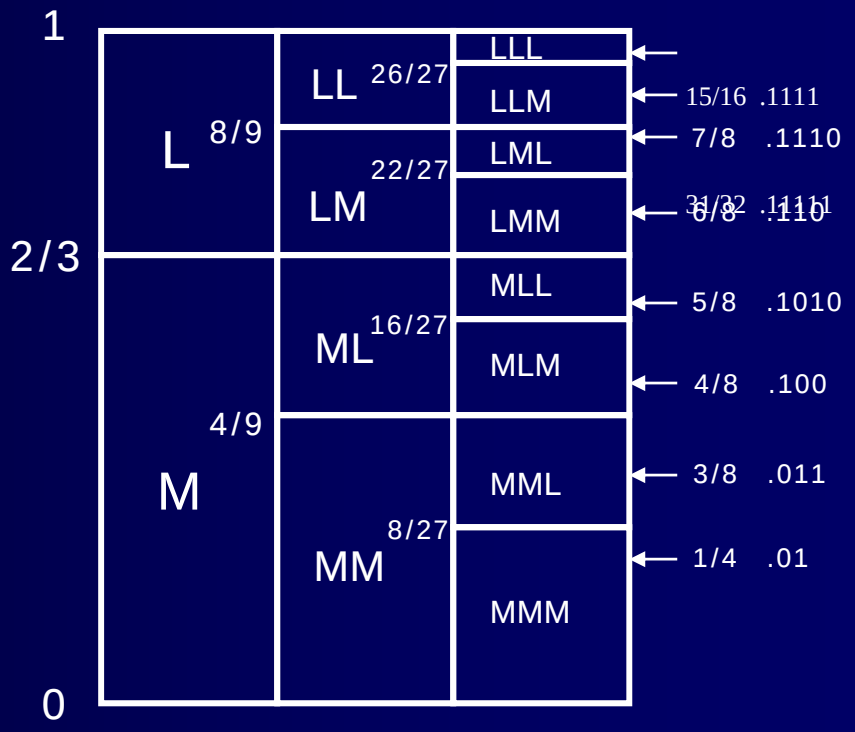
- 符号化のプロセス
- 優勢シンボル (MPS) と劣勢シンボル (LPS) の生起確率比による数直線[0,1]の多段分割
  - MPS: More probable symbol
  - LPS: Less probable symbol
- P(M)=2/3, P(L)=1/3の例
- 3個連続するシンボル系列は,“MMM”から“LLL”まで8種類の確率区間(オージェンド)に対応




数直線[0,1]の分割

# Elias Code

- Coding Process
- Multistage division of number line [0,1] by the ratio of occurrence probability
  - MPS: More probable symbol
  - LPS: Less probable symbol
- Example of P(M)=2/3, P(L)=1/3
- Three successive symbol corresponds to the probability section (Augend) from “MMM” to “LLL”




Division of Number Line [0,1]

# Eliasの符号(2)

- 算術符号化のプロセス
  - シンボル系列“MMM”は区間[0, 8/27]に相当
    “1”がMPSであれば, シンボル系列は “111”
    “0”がMPSであれば, シンボル系列は “000”
  - このオージェンドを表現する最も短い2進数は0.01
  - 小数点以下の2ビットでオージェンドを指定できる
  - 各オージェンドに対して、一意復号可能な符号を別途作成

# Elias Code(2)

- Process of Arithmetic Coding
  - Symbols “MMM” corresponds to the section [0, 8/27]

    If “1” is MPS, symbols are “111”

    If “0” is MPS, symbols are “000”

  - The minimum binary number to specify this Augend is 0.01
  - We can specify the Augend by fractional two bits
  - Create uniquely decodable code to each Augend

# Eliasの符号(3)

- 分割された各オージェンドと符号長

| Symbols | Augend | $-\mathrm{Log}_2(A)$ | Code Length (bit) |
|---|---|---|---|
| LLL | 1/27 | 4.75 | 5 |
| LLM | 2/27 | 3.75 | 4 |
| LML | 2/27 | 3.75 | 4 |
| LMM | 4/27 | 2.75 | 3 |
| MLL | 2/27 | 3.75 | 4 |
| MLM | 4/27 | 2.75 | 3 |
| MML | 4/27 | 2.75 | 3 |
| MMM | 8/27 | 1.75 | 2 |

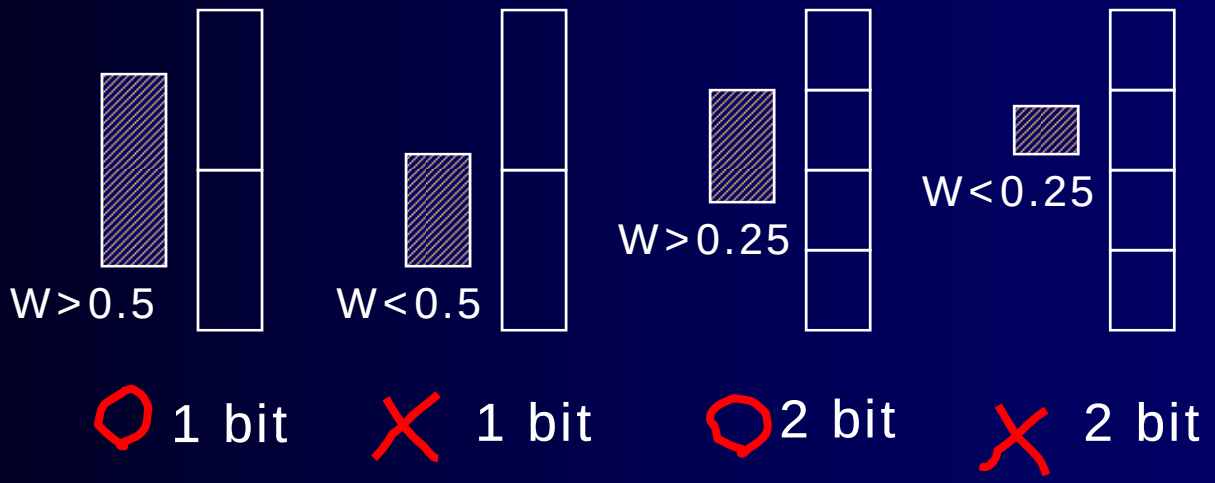
# Elias Code(3)

- Divided Augend and Code Length

| Symbols | Augend | $-\mathrm{Log}_2(A)$ | Code Length (bit) |
|---|---|---|---|
| LLL | 1/27 | 4.75 | 5 |
| LLM | 2/27 | 3.75 | 4 |
| LML | 2/27 | 3.75 | 4 |
| LMM | 4/27 | 2.75 | 3 |
| MLL | 2/27 | 3.75 | 4 |
| MLM | 4/27 | 2.75 | 3 |
| MML | 4/27 | 2.75 | 3 |
| MMM | 8/27 | 1.75 | 2 |

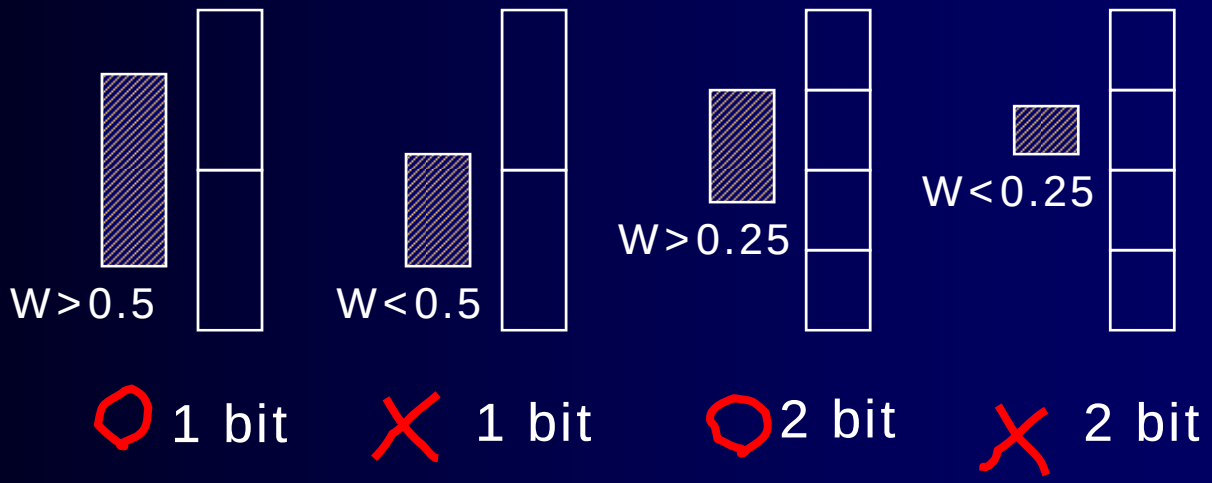
# 算術符号の原理

- 符号化のプロセス
  - MMMからLLLまでのオージェンドを計算
  - オージェンドを表現するのに最低何ビット必要か？
    - オージェンド $W$ は $M$が$a$回, $L$が$b$回であれば, $W=P(M)^aP(L)^b$



# Principle of Arithmetic Coding

- Coding Process
  - Calculate augends from MMM to LLL
  - At least how many bits required to specify Augend?
    - When $M$ is $a$-times, $L$ is $b$-times, Augend $W=P(M)^aP(L)^b$



# 算術符号の原理(2)

- オージェンド $W$の符号化に要するビット数

$$B = -\log_2 W + 1$$

- オージェンド $W$は

$$W = P(M)^a P(L)^b$$

であるから

$$B = -\log_2 (P(M)^a P(L)^b) + 1$$

# Principle of Arithmetic Coding(2)

- Bit number required for coding of Augend $W$

$$B = -\log_2 W + 1$$

- Augend $W$ is represented as

$$W = P(M)^a P(L)^b$$

Thus bit number is given by

$$B = -\log_2 (P(M)^a P(L)^b) + 1$$

# 算術符号の原理(3)

- $(a+b)$個のシンボルに対する符号長

$$
\begin{aligned}
B &= -\log_2 W + 1 \\
&= -\log_2 (P(M)^a P(L)^b) + 1 \\
&= -\log_2 P(M)^a - \log_2 P(L)^b + 1 \\
&= -a \log_2 P(M) - b \log_2 P(L) + 1 \\
&= (a+b)\left( -\frac{a}{a+b} \log_2 P(M) - \frac{b}{a+b} \log_2 P(L) \right) + 1
\end{aligned}
$$

# Principle of Arithmetic Coding(3)

- Code length for $(a+b)$ symbols

$$
\begin{aligned}
B &= -\log_2 W + 1 \\
&= -\log_2 (P(M)^a P(L)^b) + 1 \\
&= -\log_2 P(M)^a - \log_2 P(L)^b + 1 \\
&= -a \log_2 P(M) - b \log_2 P(L) + 1 \\
&= (a+b)\left( -\frac{a}{a+b} \log_2 P(M) - \frac{b}{a+b} \log_2 P(L) \right) + 1
\end{aligned}
$$

# 算術符号の原理(4)

- シンボルの個数 $N=a+b$ が無限大になれば以下が成り立つ

$$\lim_{N\to\infty}\frac{a}{N}=P(M)\quad\lim_{N\to\infty}\frac{b}{N}=P(L)$$

- このとき $N$個のシンボルに対する符号長はエントロピー$H$を用いて以下のように書ける

$$\begin{aligned}B&=(a+b)\left(-\frac{a}{a+b}\log_2 P(M)-\frac{b}{a+b}\log_2 P(L)\right)+1\\&=N\left(-P(M)\log_2 P(M)-P(L)\log_2 P(L)\right)+1\\&=NH+1\end{aligned}$$

# Principle of Arithmetic Coding(4)

- The next relation holds when symbol number $N=a+b$ becomes infinity

$$\lim_{N\to\infty}\frac{a}{N}=P(M)\qquad\lim_{N\to\infty}\frac{b}{N}=P(L)$$

- Code length for $N$ symbols can be written by entropy $H$

$$\begin{aligned}B&=(a+b)\left(-\frac{a}{a+b}\log_2 P(M)-\frac{b}{a+b}\log_2 P(L)\right)+1\\&=N\left(-P(M)\log_2 P(M)-P(L)\log_2 P(L)\right)+1\\&=NH+1\end{aligned}$$

# 算術符号の原理(5)

- 1シンボルあたりの平均符号長 $b$ は

$$b = \frac{B}{N} = H + \frac{1}{N}$$

であるから、$N$ が無限大に近づけば，$1/N$ は$0$ に近づき，平均符号長 $b$はエントロピー $H$ に漸近する

$$\lim_{N\to\infty} b = H + \lim_{N\to\infty} \frac{1}{N}$$

# Principle of Arithmetic Coding(5)

- Average code length $b$ per 1 symbol is given by

$$b = \frac{B}{N} = H + \frac{1}{N}$$

so that, when $N$ approaches to infinity, $1/N$ approaches to $0$ , and average code length $b$ approaches to entropy $H$

$$\lim_{N \to \infty} b = H + \lim_{N \to \infty} \frac{1}{N}$$

# Eliasの符号の問題点

- リアルタイム符号化
  - すべてのシンボルが出現後でないと、オージェンドを決定できないため、リアルタイム符号化が不可能
- 演算精度
  - 入力の増加と共に、オージェンドを計算するために、積を求める必要が生じる
  - シンボル系列長の増加と共に高い演算精度が必要となるため、実現不可能

# Problem of Elias Code

- Realtime coding
  - It is impossible to code in realtime since we cannot specify Augend after the appearance of all symbol.
- Operational accuracy
  - Multiplication is necessary to calculate Augend, whenever the input symbol is increased
  - It is impossible to realize since unlimited high operational accuracy is necessary according to the increase of symbol length

# L-R符号化

- Eliasの符号の問題点の解決
  - リアルタイム性

    符号をオージェンド $[L,H)$ の下側の値 $L$ にとる

    桁上がりによるビット反転の波及をビットスタッフィングで防止

  - 演算精度
    - LPSの生起確率を $2^{-Q}$ で近似
    - 浮動小数点演算を行い、演算を有限桁で打ち切り

# L-R Coding

- Solution to the problem of Elias Coding
  - Real-time operation
    - Set code at the lower end $L$ of Augend $[L,H)$
    - Prevent propagation problem of bit reversal caused by carry using bit stuffing
  - Operational accuracy
    - Approximate LPS's occurrence probability by $2^{-Q}$
    - Floating point operation, cut off operation at finite digit

# L-R符号化アルゴリズム

- 符号化のプロセス
  - 初期設定
    - VビットのレジスタCおよびレジスタAを用意
    - Cにはすべて0，Aにはすべて1を代入

      C ← 00000… 0 (V bit) ：符号の初期値

      A ← 11111… 1 (V bit) ：オージェンド
  - シンボル$x_i$ の符号化
    - Aの値を LPS(＝1)，MPS(＝0)の生起確率比 $p_1{:}p_0$ に分割
    - 分割した値をレジスタ$A_0$，$A_1$に格納

# L-R Coding Algorithm

- Coding Process
  - Initialization
    - Prepare register C and A with V bit
    - Insert all 0 to C, all 1 to A

      C ← 00000… 0 (V bit) : Initial value of code

      A ← 11111… 1 (V bit) : Augend
  - Coding of symbol $x_i$
    - Divide A based on occurrence probability ratio $p_1$:$p_0$ of LPS(=1) and MPS(=0)
    - Store divide value to register $A_0$ and $A_1$

# L-R符号化アルゴリズム(2)

- 符号化のプロセス
  - 分割したレジスタ
    - $A_1 \leftarrow A\ p_1$
    - $A_0 \leftarrow A\ p_0$
  - LPSシンボルの生起確率 $p_1$ を $2^{-Q}$ で近似（$Q$:整数、スキューと呼ぶ）し、乗算をビットシフト演算に置き換える
    - $A_1 \leftarrow A\ 2^{-Q}$
    - $A_0 \leftarrow A - A_1$
  - 演算レジスタ長はVビットであるから、スキューの最大値はV-1

# L-R Coding Algorithm(2)

- Coding Process
  - Divided register
    - $A_1 \leftarrow A\ p_1$
    - $A_0 \leftarrow A\ p_0$
  - Approximate occurrence probability $p$ of LPS symbol by $2^{-Q}$ (Q:integer, Called Sque), multiplication is replaced by bit-shift operation
    - $A_1 \leftarrow A\ 2^{-Q}$
    - $A_0 \leftarrow A - A_1$
  - Operation register length is V bit, so that the maximum value of Sque is V-1

# L-R符号化アルゴリズム(3)

- 符号化のプロセス
  - 符号の生成
    - 符号はオージェンドの最下値に設定したため、入力シンボル xi の値によって次式で計算

      $x_i=0$ のとき　レジスタCはそのまま

      $x_i=1$ のとき　C ← C + $A_0$

  - オージェンドの更新
    - 次の入力に備えてオージェンドを以下のように更新

      $x_i=0$ のとき　A ← $A_0$

      $x_i=1$ のとき　A ← $A_1$

# L-R Coding Algorithm(3)

- Coding Process
  - Generation of code
    - Code was set to the minimum value of Augend, so that code can be obtained for each input symbol xi

      When $x_i=0$ register C does not change

      When $x_i=1$ C ← C + $A_0$
  - Renewal of Augend
    - For preparation of the next input, Augend is renewed as follows

      When $x_i=0$ A ← $A_0$

      When $x_i=1$ A ← $A_1$

# L-R符号化アルゴリズム(4)

- オージェンドの再正規化(Re-Normalize)
  - 入力シンボル系列が増加するに従い、オージェンドの値は減少
  - ところが、レジスタ長Vは一定
  - 対策として、レジスタAの MSB (Most Significant Bit) が0ならば、1になるまで左シフト
  - 同様にCも同じ量だけ左シフト
  - シフトの結果、オーバフローしたビットが符号語となる

A ← 000011... 1 (V bit)
An ← 11... 11... 1 (V bit)
C のMSB側 4 bit が符号語

# L-R Coding Algorithm(4)

- Re-normalization of Augend
  - Augend value decreases in accordance with the increase of input symbols
  - However, register length V is constant
  - Remedy is to operate left shift of register A if MSB is 0, unitl it will be 1
  - Similarly, make left-shift for C as the same amount
  - Overflowed bits are codeword by this shifting

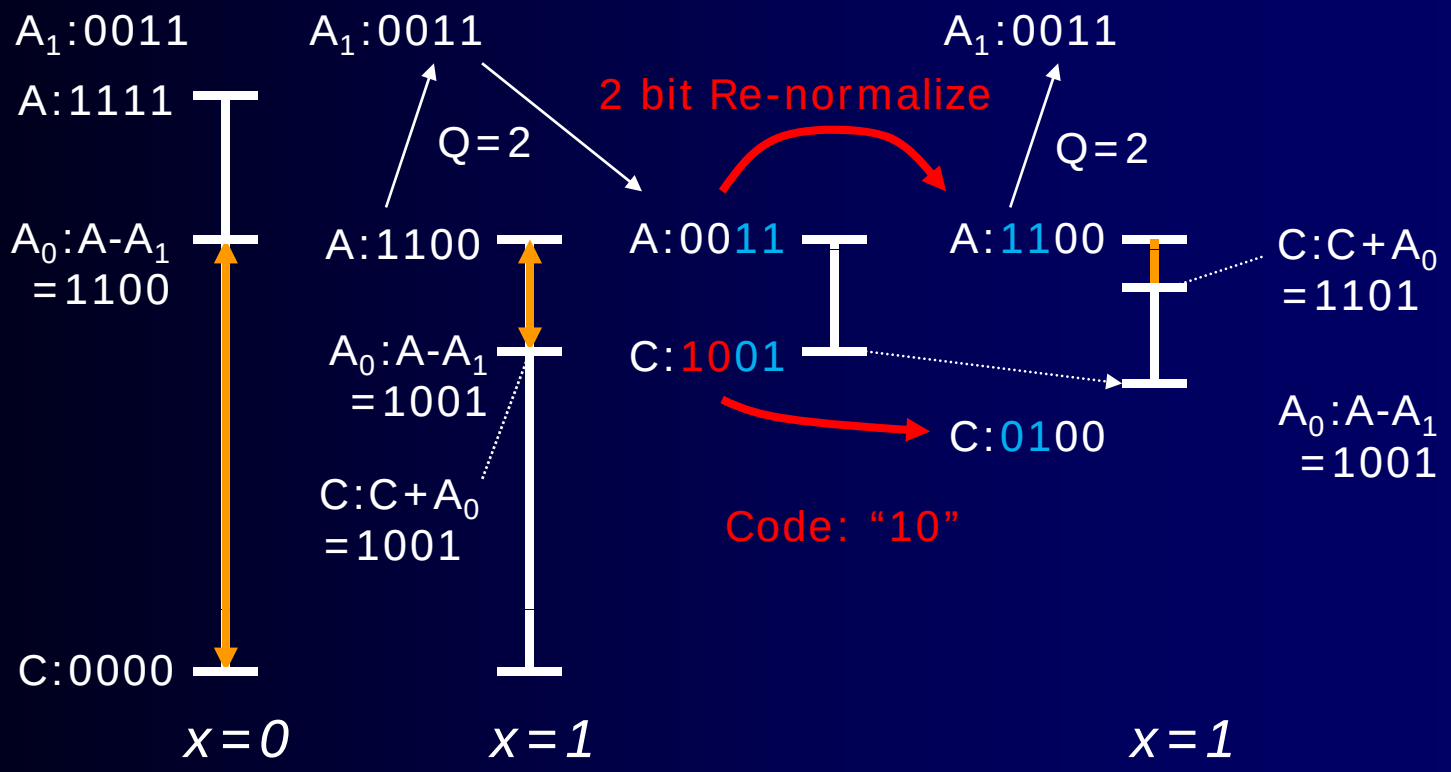
A ← 000011... 1 (V bit)

An ← 11... 11... 1 (V bit)

MSB side of C (4 bit) is codeword

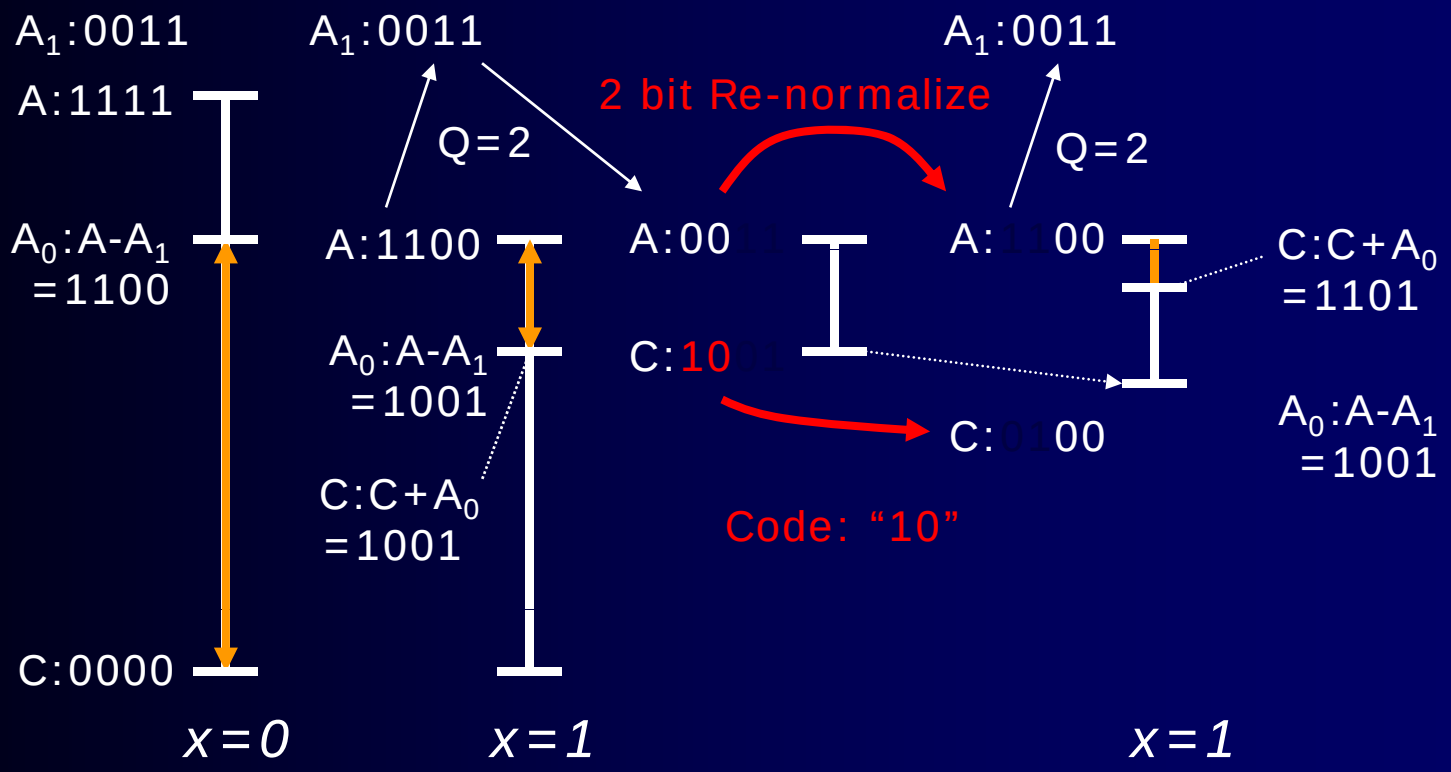
# L-R符号化の計算例

- 計算例 (V=4, Q=2, *x=011...*)



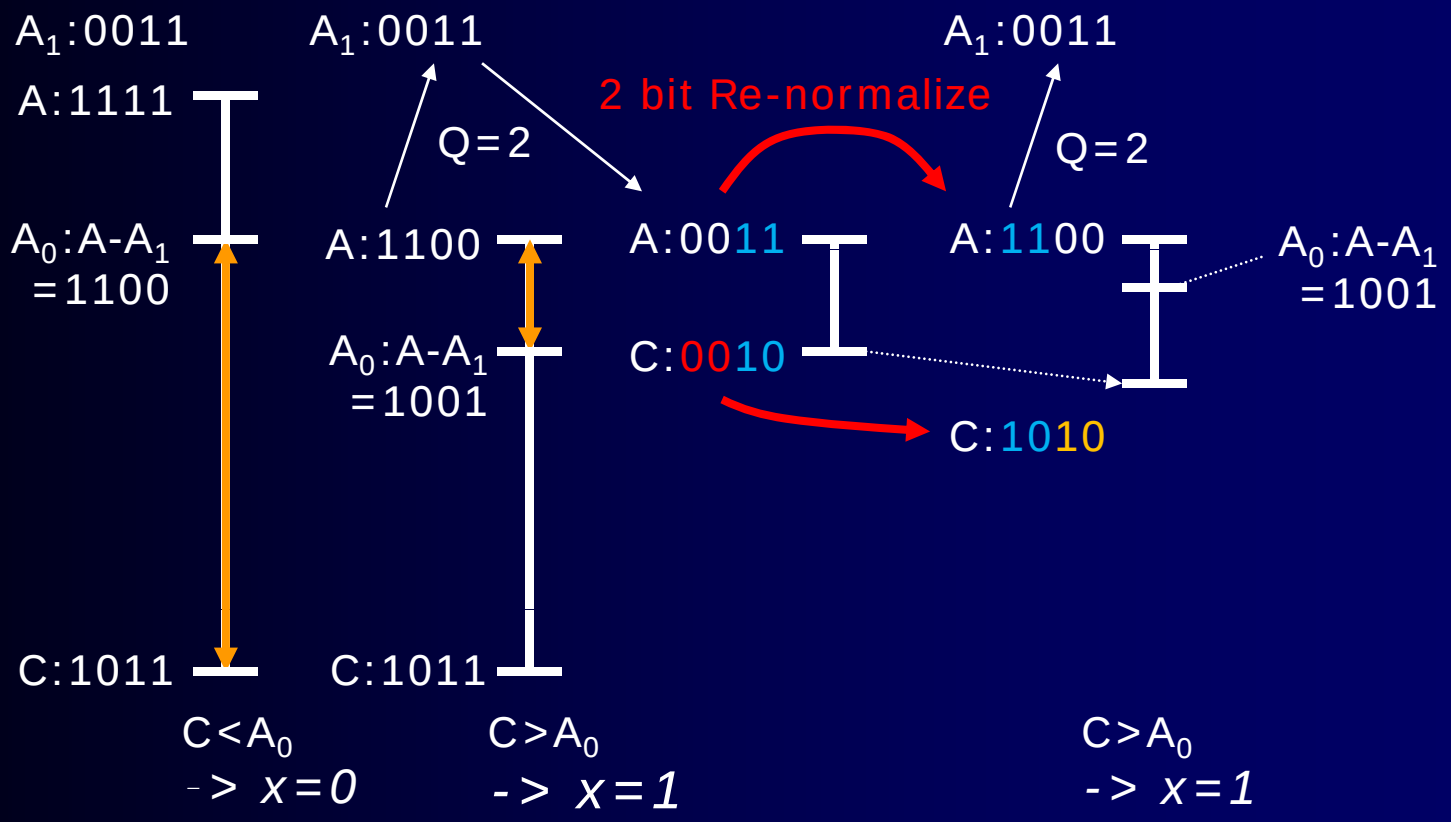
# Example of L-R Coding

- Example (V=4, Q=2, *x=011*...)



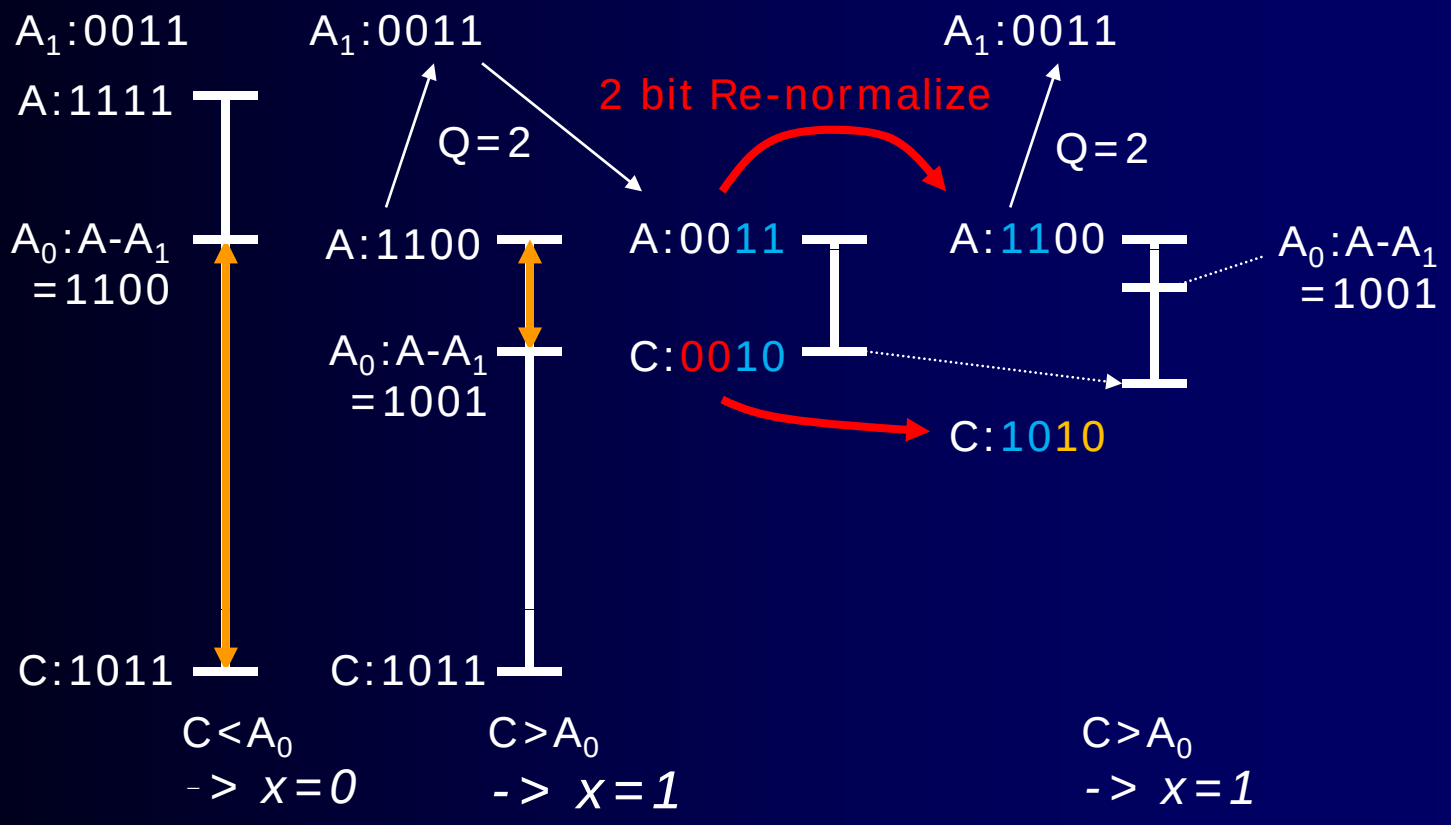
# L-R復号の計算例

- 計算例 (V=4, Q=2, Code=101110... )



# Example of L-R Decoding

- Example (V=4, Q=2, Code=101110... )



# 問題

- L-R算術符号化において、V=4, Q=2, x=100..., であるとき、どのように A-レジスタ, A0-レジスタ, A1-レジスタ, C-レジスタの値は変化するかを示せ.

# Quiz

- When V=4, Q=2, x=100…, how do values of A-register, A0-register, A1-register, C-register of L-R arithmetic coding change?

# L-R符号化の平均符号長

- 平均符号長

$$L = -p_0 \log_2 (1 - 2^{-Q}) - p_1 \log_2 2^{-Q}$$
$$= -p_0 \log_2 (1 - 2^{-Q}) + p_1 Q$$

- エントロピー

$$H = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

# Average Code Length of L-R Coding

- Average Code Length

$$L = -p_0 \log_2 (1 - 2^{-Q}) - p_1 \log_2 2^{-Q}$$
$$= -p_0 \log_2 (1 - 2^{-Q}) + p_1 Q$$

- Entropy

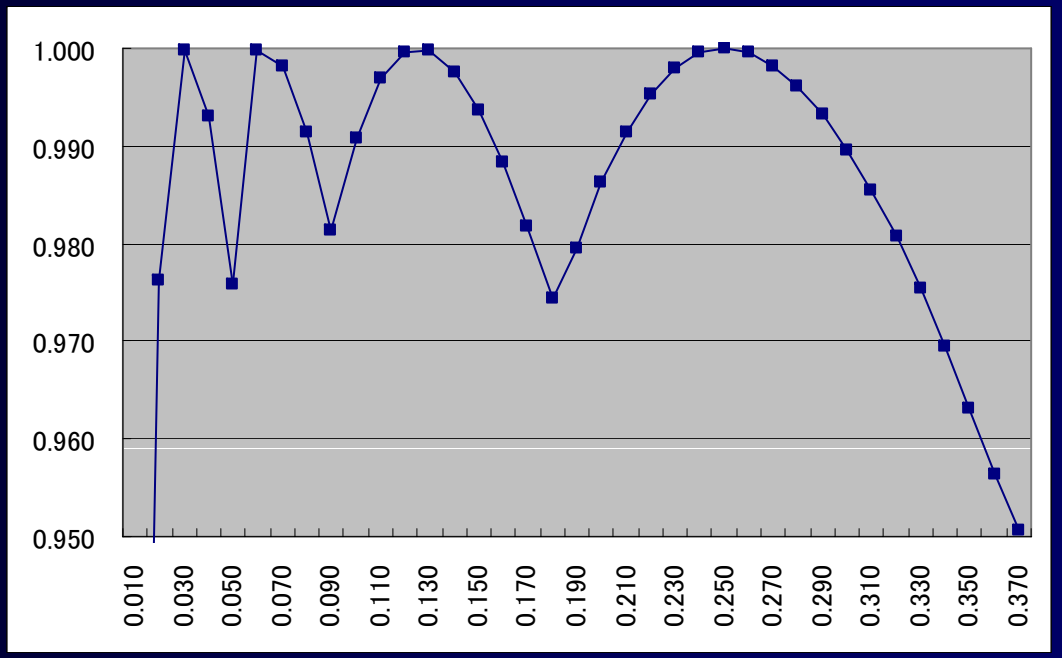
$$H = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

# L-R符号化の符号化効率

- L-R算術符号化の符号化効率

H/L

H:エントロピー
L:平均符号長




$P_1$

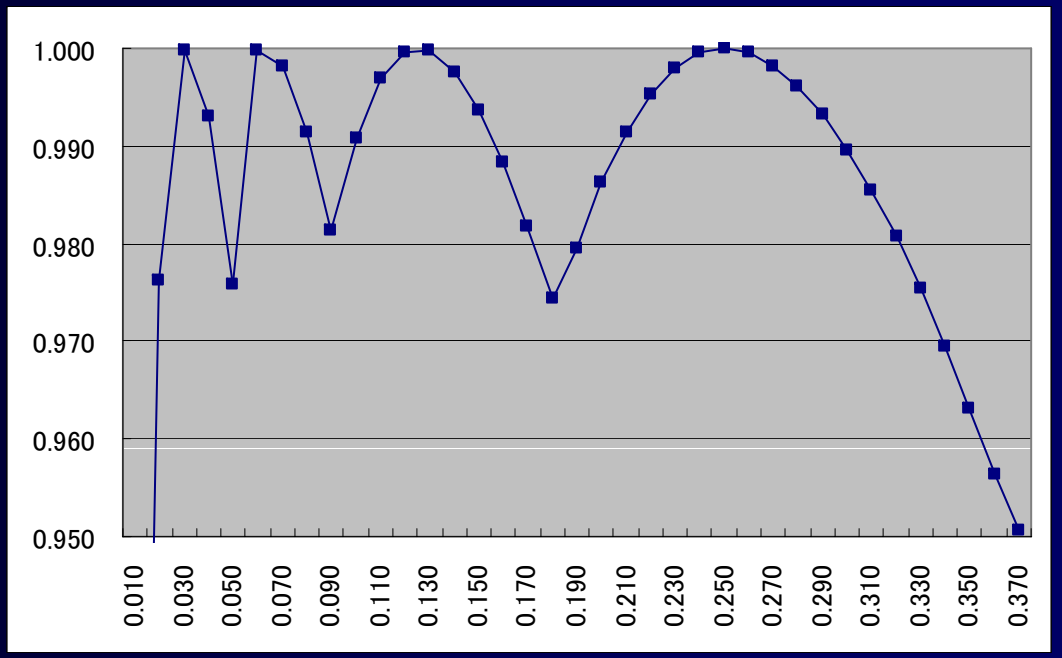
符号化効率

# Efficiency of L-R Coding

- Efficiency of L-R Arithmetic Coding

H/L

H:Entropy
L:Average
Code
Length




$P_1$

Efficiency

# 動的算術符号

- Q-coder
  - スキューの動的変化
  - 演算の簡単化
  - エラー伝播の防止
- MELCODE
  - 適応ブロック符号化
  - 算術符号と同様の構成が可能
- QM-coder
  - Q-coderとMELCODEを合わせた符号化
  - JPEG, JBIGで使用

# Adaptive Arithmetic Coding

- Q-coder
  - Dynamic change of Sque
  - Simplification of operation
  - Prevent error propagation
- MELCODE
  - Adaptive block coding
  - Similar structure with arithmetic coding is possible
- QM-coder
  - Q-coder and MELCODE are combined
  - Used by JPEG, JBIG