アルゴリズム的ランダムネス入門

— C言語によるアルゴリズム的ランダムネス —

只木孝太郎

中央大学 研究開発機構

Supported by KAKENHI (23340020) and KAKENHI (24540142)

本講演で何をするか

C言語によるアルゴリズム的ランダムネス

本講演で何をするか

C言語によるアルゴリズム的ランダムネス

基礎コースの知見でアルゴリズム的ランダムを理解する。
発展コースの各講義のための基礎知識を提供し、
基礎コースと発展コースのギャップを埋める。

アルゴリズム的ランダムネスとは？

無限２進列の分類

アルゴリズム的ランダムネスは、個々の無限２進列に対し、**ランダムなもの**と**ランダムでないもの**への分類を可能にする。

000000000000000000000000000000000000000000 ・・・・・・・・・・・・・・

これはランダムでない。

010101010101010101010101010101010101010101 ・・・・・・・・・・・・・・

これはランダムでない。

010110010010110010100110101110011011100 1 ・・・・・・・・・・・・・・

これはランダムである。

110010010000111111011010101000100010000 1 ・・・・・・・・・・・・・・

これはランダムでない。

無限２進列の分類

アルゴリズム的ランダムネスは、個々の無限２進列に対し、**ランダムなもの**と**ランダムでないもの**への分類を可能にする。

000000000000000000000000000000000000000000 ・・・・・・・・・・・・・・

これはランダムでない。

010101010101010101010101010101010101010101 ・・・・・・・・・・・・・・

これはランダムでない。

010110010010110010100110101110011011100 1 ・・・・・・・・・・・・・・

これはランダムである。

110010010000111111011010101000100010000 1 ・・・・・・・・・・・・・・

πの２進展開　　**これはランダムでない。**

アルゴリズム的ランダムネスにおけるランダム性の定義の方法

- 圧縮不可能性に基づく定義:

  1-ランダム性（Chaitinランダム性)、その変種

- 測度論的定義:

  Martin-Löfランダム性、その変種

- 賭博戦略に基づく定義

- 微分可能性に基づく定義

- ・・・・・・・・・・・・

アルゴリズム的ランダムネスにおけるランダム性の定義の方法

- 圧縮不可能性に基づく定義:

  1-ランダム性（Chaitinランダム性)、その変種

- 測度論的定義:

  Martin-Löfランダム性、その変種

- 賭博戦略に基づく定義

- 微分可能性に基づく定義

- ・・・・・・・・・・・・

圧縮不可能性に基づくランダムネスの定義

はじめに、無限２進列ではなく、
有限２進列の圧縮を考える。

準備: 有限２進列

- $\{0,1\}^* := \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$
  有限２進列の集合 (the set of finite binary strings)
  ここで、εは空列(empty string)を表す.

- 任意の$s \in \{0,1\}^*$に対し、$|s|$はsの長さ(length)を表す.

例えば　$|010| = 3$,　$|\varepsilon| = 0$.

基礎コースの入門講座「計算可能性の理論」では、計算可能性を議論する対象として、自然数から自然数への関数や自然数の集合を考えたが、アルゴリズム的ランダムネスでは、有限２進列から有限２進列への関数や有限２進列の集合が重要な役割を果たす。基礎コースで学んだ知見を生かすため、有限２進列と自然数を次のように同一視して、基礎コースの概念・方法をこの講義で活用する。

$\{0,1\}^*$	:	ε,	0,	1,	00,	01,	10,	11,	000,	001,	010,	011,	...
$\mathbb{N}$	:	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	...

即ち、$\varphi\colon \{0,1\}^* \to \mathbb{N}$で$\varphi(s) = (1s)_2 - 1$なる写像を考え、有限２進列$s$と自然数$\varphi(s)$を同一視する。ここで、$(1s)_2$は２進表示が$1s$となる整数である。

有限２進列：長さの計算

例えば　　$\varphi(s) = (1s)_2 - 1$ なので、

$$\varphi(\varepsilon) = (1)_2 - 1 = 1 - 1 = 0$$
$$\varphi(0) = (10)_2 - 1 = 2 - 1 = 1$$
$$\varphi(1) = (11)_2 - 1 = 3 - 1 = 2$$

などが成り立つ。

例えば、“ s から $|s|$ は計算可能である ”即ち、１変数全域関数 strlen : $\{0,1\}^* \to \mathbb{N}$ を $\mathrm{strlen}(s) = |s|$ で定義すると、これは次の１入力関数プログラムで計算される。

```
strlen(x){                  /*  入力は有限２進列s、即ち、自然数x=1s-1だとする。 */
int len;
  x = x + 1;                /*  1s-1を1sに戻す。 */
  len = 0;                  /*  lenを初期化する。 */
  while (x != 1) {          /*  1sの1が一の位に移動すれば脱出。 */
    len ++;
    x = x / 2;              /*  xを1ビット右にシフトする。 */
  }
  return(len);
}
```

有限２進列：連接の計算

$\varphi(s) = (1s)_2 - 1$ なので、

連接 (concatenation) st について次が成り立つ。

$$\varphi(st) = \varphi(s)2^{|t|} + \varphi(t).$$

が成り立つ。

２変数全域関数 concatenate: $\{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$ を

$$\mathsf{concatenate}(s,t) = st$$

で定義すると、これは次の２入力関数プログラムで計算される。

```
concatenate(x,y){
  loop (strlen(y)) {        /* 以下をstrlen(y)回繰り返す。 */
    x = 2 * x;              /* xを1ビット左にシフトする。 */
  }
  return(x+y);
}
```

Plain Descriptive Complexity

Plain Descriptive Complexity

定義 [Machine] machineとは1変数計算可能部分関数のことである（machineと言うとき、有限2進列から有限2進列への関数と見なしている）。 □

定義 任意のmachine Mと任意の$x \in \{0,1\}^*$に対し、$C_M(x)$を次式で定義する。

$$C_M(x) = \min\{|\sigma| \mid \sigma \in \{0,1\}^* \ \&\ M(\sigma) = x\}.$$

□

定義 [Optimality] Rをmachineとする。任意のmachine Mに対し、ある自然数dが存在し、任意の有限2進列xに対し、次が成り立つとき、Rはoptimalであると言う。

$$C_R(x) \leq C_M(x) + d.$$

□

定理 optimal machineが存在する。 （後で証明する）

□

定義 [Plain Descriptive Complexity] optimal machine Rを任意に選んで固定し、次式で$C(x)$を定義する。これをxのplain descriptive complexityと呼ぶ。

$$C(x) = C_R(x).$$

□

Plain Descriptive Complexity

注意 RとR'をともにoptimal machineとすると、optimal性から、ある自然数dが存在し、任意のxに対し、次が成り立つ。

$$|C_R(x) - C_{R'}(x)| \leq d.$$

□

定理 ある自然数dが存在し、任意の有限２進列xに対し、次が成り立つ。

$$C(x) \leq |x| + d.$$

証明)

１変数全域関数Mを$M(\sigma) = \sigma$で定義する。Mは次の１入力関数プログラムで計算されるので、machineである。

```
M(x){
  return(x);
}
```

一方、optimalityから、ある自然数dが存在し$C(x) \leq C_M(x) + d$が成り立ち、また

$$C_M(x) = \min\{|\sigma| \mid M(\sigma) = x\} = \min\{|x|\} = |x|$$

が成り立つことから、定理は成り立つ。

□

準備: e番目のk変数計算可能部分関数

定理 [基礎コース教科書p.49の定理 3.4.1] 任意の自然数kと任意のk変数計算可能部分関数fに対して、ある自然数pが存在して、次が成り立つ。

$$f(x_1, \ldots, x_k) = \mathsf{comp}(p, \langle x_1, \ldots, x_k \rangle).$$

□

定理 [基礎コース教科書p.49の定理 3.4.2] compは計算可能である。 □

定義 kとeは自然数とする。k変数部分関数Φ_e^kを次式で定義する。

$$\Phi_e^k(x_1, \ldots, x_k) = \mathsf{comp}(e, \langle x_1, \ldots, x_k \rangle).$$

□

定理 任意の自然数kに対して次が成り立つ。

(i) 任意のk変数計算可能部分関数fに対して、ある自然数eが存在して、次が成り立つ。

$$f(x_1, \ldots, x_k) = \Phi_e^k(x_1, \ldots, x_k).$$

(ii) 任意のeに対してΦ_e^kはk変数計算可能部分関数である。 □

全てのk変数計算可能部分関数は、$\{\Phi_e^k\}_{e \in \mathbb{N}}$で過不足なく尽されるので（重複はあるが）、
Φ_e^kをe番目のk変数計算可能部分関数と呼び、eをΦ_e^kのindexと呼ぶ。

以下ではΦ_e^1が重要な役割を果し、これを単にΦ_eと書く。

Plain Descriptive Complexity

定理 [再掲] optimal machineが存在する。

証明)

1変数部分関数Rを$R(\sigma 10^e) = \Phi_e(\sigma)$で定義する（この関数は記号1を含まない有限2進列に対しては定義されない)。Rは次の1入力関数プログラムで計算されるので、machine（1変数計算可能部分関数）である。

```
R(x){
int e;
  x ++;
  e = 0;
  while (x % 2 != 1) {
    e ++;
    x = x / 2;
  }
  x = x / 2;
  x --;
  return(comp(e,pair(x,0)));
}
```

Plain Descriptive Complexity

定理 [再掲] optimal machineが存在する。

証明（続き）)

Rがoptimalであることは次のようにしてわかる。
Mを任意のmachineとする。このときある自然数 (index) eが存在し、$M(\sigma) = \Phi_e(\sigma)$が成り立つ。このとき、任意の$x$に対し、$\sigma_0$を$M(\sigma_0) = x$となる最も長さの短い有限2進列とすると、

$$C_M(x) = |\sigma_0| .$$

が成り立つ。
一方、

$$R(\sigma_0 10^e) = \Phi_e(\sigma_0) = M(\sigma_0) = x$$

が成り立つので、

$$C_R(x) \leq |\sigma_0 10^e| = |\sigma_0| + 1 + e = C_M(x) + e + 1$$

が成り立つ。従ってRはoptimalである。 □

準備: 無限２進列

- 無限２進列の集合（Cantor space）を$2^{\mathbb{N}}$で表す。

- 任意の$X \in 2^{\mathbb{N}}$に対し、Xの最初のnビット（Xの長さnのprefix）を$X\restriction_n$で表す。

例えば、　$X = 10110010\cdots \in 2^{\mathbb{N}}$のとき、$X\restriction_5 = 10110$であり、$X\restriction_0 = \varepsilon$である。

Plain Descriptive Complexityではランダム性は定義できない。

定義 [$C(x)$によるランダム性の定義の試み] $X \in 2^{\mathbb{N}}$について、ある自然数dが存在し、任意の自然数nに対して、次が成り立つとき、Xは“ ランダム ”であると言われる。

$$n - d \leq C(X\restriction_n).$$

□

定理 [Martin-Löf 1971] 任意の$X \in 2^{\mathbb{N}}$に対し、$C(X\restriction_n) - n$は下に有界でない。

証明） 1変数全域関数Mを$M(\sigma) = \varphi^{-1}(|\sigma|)\sigma$で定義する。$M$は明らかにmachineである。さて、$m$を任意の自然数とすると、$X\restriction_m = \varphi^{-1}(k)$となる自然数$k$が存在する。

$$X\restriction_{m+k} = \varphi^{-1}(k)\sigma$$

となるように長さkのσを選ぶ。すると$M(\sigma) = X\restriction_{m+k}$が成り立つ。このとき、

$$C(X\restriction_{m+k}) \leq C_M(X\restriction_{m+k}) + O(1) \leq |\sigma| + O(1) = k + O(1)$$

が成り立つ。従って、

$$C(X\restriction_{m+k}) \leq (m + k) - m + O(1)$$

であり、ここでmは任意なので、$C(X\restriction_n) - n$は下に有界でない。 □

Prefix-Free Complexity

準備: Prefix-Free集合

- 有限２進列$s, t, u \in \{0,1\}^*$に対し、$st = u$が成り立つとき、sはuの接頭語(prefix)と呼ばれる。

- 集合$P \subset \{0,1\}^*$に対し、Pがprefix-freeであるとは、任意の異なる$s, t \in P$に対し、sがtの接頭語(prefix)にはならないことを言う.

For example　$\{0, 10\}$: prefix-free.
$\{0, 01\}$: not prefix-free.

- prefix-free集合は有限集合の場合もあるし、無限集合の場合もある.

- 任意のprefix-free集合$P \subset \{0,1\}^*$に対し、次が成り立つ:

$$\sum_{s \in P} 2^{-|s|} \leq 1. \qquad \text{(Kraftの不等式)}$$

準備: Prefix-Free集合

自然数上の２変数全域関数is_prefixを次のように定義する。

(i) 有限二進列s, tに対して、sがtのprefixとなっている場合、is_prefix$(s, t) = 1$.

(ii) それ以外の場合、is_prefix$(s, t) = 0$. □

is_prefixは次の２入力関数プログラムで計算され、計算可能である。

```
is_prefix(s,t){
  int u;
  u = 0;
  loop (t) {
    if (t == concatenate(s,u))
      return(1);
    u ++;
  }
  return(0);
}
```

準備: Prefix-Free集合

- 自然数上の２変数全域関数is_prefix-freeを次のように定義する。

(i) 自然数$a = \langle a_1, \ldots, a_n \rangle$に対して、$\{a_1, \ldots, a_n\}$がprefix-free集合となっている場合、is_prefix-free$(a) = 1$.

(ii) それ以外の場合、is_prefix-free$(a) = 0$. □

is_prefix-freeは、is_prefixを呼び出す１入力関数プログラムで計算されるので、計算可能である。

Prefix-Free Complexity

定義 [Prefix-Free Machine] 定義域がprefix-free集合となっているmachine（1変数計算可能部分関数）をprefix-free machineと呼ぶ。 □

定義 任意のprefix-free machine Mと任意の$x \in \{0,1\}^*$に対し、$K_M(x)$を次式で定義する。

$$K_M(x) = \min\{|\sigma| \mid M(\sigma) = x\}.$$ □

定義 [Optimalty] Rをprefix-free machineとする。任意のmachine Mに対し、ある自然数dが存在し、任意の有限2進列xに対し、次が成り立つとき、Rはoptimalであると言う。

$$K_R(x) \leq K_M(x) + d.$$ □

定理 optimal prefix-free machineが存在する。 （後で証明する） □

定義 [Prefix-Free Complexity] optimal prefix-free machine Uを任意に選んで固定し、次式で$K(x)$を定義する。これをxのprefix-free complexityと呼ぶ。

$$K(x) = K_U(x).$$ □

Prefix-Free Complexity

注意 RとR'をともにoptimal prefix-free machineとすると、optimalityから、ある自然数dが存在し、任意のxに対し、次が成り立つ。

$$|K_R(x) - K_{R'}(x)| \leq d.$$

□

Prefix-Free Complexity

以下、$K(x)$の定義に用いたoptimal prefix-free machineをUとおく。

定義 有限2進列xに対し、$U(\sigma) = x$なる有限2進列σのうち、自然数と見たときに最小となるσをx^*と書く。 □

このとき、明らかに$K(x) = |x^*|$が成り立つ。

定理 $K(x) \leq |x| + K(|x|) + O(1)$.

証明)

1変数部分関数Mを次のように定義する。

(i) 有限2進列σに対し、$U(p) = |x|$なる有限2進列p, xが存在して$\sigma = px$が成り立つ場合、$M(\sigma) = x$.

(ii) それ以外の場合、$M(\sigma)$は定義されない。 □

このとき、Mはprefix-free machineであることを示すことができる。

さて、任意の有限2進列xに対して、$U(|x|^*) = |x|$であるので、$M(|x|^* x) = x$が成り立つ。また、$K(|x|) = ||x|^*|$が成り立つ。一方、ある自然数dが存在し$K(x) \leq K_M(x) + d$が成り立ち、

$$K(x) \leq K_M(x) + d \leq ||x|^* x| + d = |x| + ||x|^*| + d = |x| + K(|x|) + d.$$

□

タイマー付き万能計算

次の定理の証明の準備をする。

定理 [再掲] optimal prefix-free machineが存在する。 □

- 自然数上の３変数全域関数halt_beforeを次のように定義する。

(i) executable$(p, x) = 1$であり、pが表すジャンププログラムPの入力変数にxが表す自然数列の各要素の値をセットして実行開始すると、tステップ以内にすなわちt回以下の文の実行によって終了する場合、halt_before$(p, x, t) = 1$.

(ii) それ以外の場合、halt_before$(p, x, t) = 0$. □

定理 [基礎コース教科書 第４章 章末問題3] halt_beforeは３変数計算可能全域関数である。

証明)

基礎コース教科書の定理3.4.2「compは計算可能である」の証明にある、compの関数プログラム（図3.6）を修正すればよい。 □

タイマー付き万能計算

```
domain_before(p,t){
int list, x;
  list = 0;
  x = 0;
  while (x <= t) {
    if (halt_before(p,pair(x,0),t) == 1)
      list = pair(x,list);
    x ++;
  }
  return(list);
}
```

定理　上記の関数プログラムで計算される２変数全域関数domain_beforeは次の性質を持つ。domain_before(p,t)が表す自然数列の各要素を集めた自然数の集合を$S(p,t)$とおく。このとき、

(i)　$S(p,t)$の元は全てt以下。

(ii)　$S(p,t) \subset S(p,t+1)$.

(iii)　$\bigcup_{t=0}^{\infty} S(p,t) = \{x \mid \mathsf{comp}(p,\langle x \rangle)$は定義される$\}$.　□

なお、listの自然数列に、同じ自然数は２度現れないことに注意。

列操作関数

- 自然数上の２変数全域関数is_elementを次のように定義する。

(i) 自然数$a = \langle a_1, \ldots, a_n \rangle$に対して、$x \in \{a_1, \ldots, a_n\}$の場合、is_element$(a, x) = 1$.

(ii) それ以外の場合、is_element$(a, x) = 0$. □

明らかに、is_elementは計算可能である。

Prefix-Free Complexity

```
prefix_comp(p,x){
int stage, list;
  stage = 0;
  list = 0;
  while (is_prefix-free(list) == 1) {
    if (is_element(list,x))
      return(comp(p,pair(x,0));
    stage ++;
    list = domain_before(p,stage);
    }
  }
  while (0 == 0);
}
```

定理　上記の関数プログラムで計算される２変数部分関数 prefix_comp は次の性質を持つ。

(i) 任意の prefix-free machine M に対して、ある自然数 e が存在し、
$M(\sigma) = \text{prefix\_comp}(e, \sigma)$ が成り立つ。

(ii) 任意の自然数 e に対し、$\{\sigma \mid \text{prefix\_comp}(e, \sigma)$ は定義される $\}$ は prefix-free 集合である。 □

列操作関数

- 自然数上の１変数部分関数ext_codeを次のように定義する。

(i) 有限２進列が１を含んでいる場合、$\sigma = 0^e 1\rho$とおくと、$\mathsf{ext\_code}(\sigma) = e$.

(ii) それ以外の場合、$\mathsf{ext_code}(\sigma)$は定義されない。 □

- 自然数上の１変数部分関数ext_dataを次のように定義する。

(i) 有限２進列が１を含んでいる場合、$\sigma = 0^e 1\rho$とおくと、$\mathsf{ext\_data}(\sigma) = \rho$.

(ii) それ以外の場合、$\mathsf{ext_data}(\sigma)$は定義されない。 □

明らかに、ext_code, ext_dataは計算可能である。

Prefix-Free Complexity

定理 [再掲] optimal prefix-free machineが存在する。

証明)

1変数部分関数Rを$R(0^e1\sigma) = \text{prefix_comp}(e, \sigma)$で定義する（この関数は記号1を含まない有限2進列に対しては定義されない)。

Rは次の1入力関数プログラムで計算されるので、machine（1変数計算可能部分関数）である。

```
R(x){
int e, s;
  e = ext_code(x);
  s = ext_data(x);
  return(prefix_comp(e,s));
}
```

また、任意の自然数eに対し、$\{\sigma \mid \text{prefix_comp}(e, \sigma)$は定義される$\}$はprefix-free集合であるので、$R$はprefix-free machineである。

Rがoptimalであることは、optimal machineの存在証明と同様に証明できる。 □

1-ランダムネス

定義 [1-ランダムネス] $X \in 2^{\mathbb{N}}$ について、ある自然数 d が存在し、任意の自然数 n に対して、次が成り立つとき、X は1-ランダム（Chaitinランダム）である言われる。

$$n - d \leq K(X\restriction_n).$$

□

ChaitinのΩ

定義 [ChaitinのΩ] 任意のprefix-free machine M に対し、Ω_M を次式で定義する。

$$\Omega_M = \sum_{M(\sigma)\downarrow} 2^{-|\sigma|}.$$

□

（注意）Kraftの不等式により、Ω_M は収束し、$\Omega_M \leq 1$ を満たす。

定理 [Chaitin 1975] 任意のoptimal prefix-free machine R に対し、Ω_R の2進展開は1-ランダムである。 □

ChaitinのΩ

定理 [Chaitin 1975, 再掲] 任意のoptimal prefix-free machine Rに対し、Ω_Rは1-ランダムである。

証明)

Rは1変数計算可能部分関数なので、ある自然数eが存在して$R(\sigma) = \mathrm{comp}(e, \langle\sigma\rangle)$が成り立つ。

自然数上の2変数全域関数excessを次のように定義する。

(i) 自然数$a = \langle a_1, \ldots, a_n\rangle$と$s$に対して、$0.s < \sum_{i=1}^{n} 2^{-|a_i|}$となっている場合、$\mathrm{excess}(a, s) = 1$. ここで、$s$と$a_1, \ldots, a_n$を有限2進列とみなしている。

(ii) それ以外の場合、$\mathrm{excess}(a, s) = 0$.

excessは、明らかに計算可能である。

ChaitinのΩ

定理 [Chaitin 1975, 再掲] 任意のoptimal prefix-free machine Rに対し、Ω_Rは1-ランダムである。

証明（続き）） 次の関数プログラムを考えよう。

```
f(s){
int stage, list, x, i;
  stage = 1;
  while (0 == 0) {
    list = domain_before(e,stage);
    if (excess(list,s) == 1) {
      i = 1;
      loop (length(list)) {
        list = replace(list,i,comp(e,pair(element(list,i),0)));
        i ++;
      }
      x = 0;
      while (is_element(list,x) != 1) {
        x ++;
      }
      return(x);
    }
    stage ++;
  }
}
```

ChaitinのΩ

定理 [Chaitin 1975, 再掲] 任意のoptimal prefix-free machine Rに対し、Ω_Rは1-ランダムである。

補題 任意の1変数計算可能部分関数fに対して、$K(f(x)) \leq K(x) + O(1)$が成り立つ。 □

証明（続き））

入力を$\Omega_R{\restriction}_k$としたとき、`list`にi番目に加わった有限2進列をa_iとすると、`if`の条件が真になったとき、$0.\Omega_R{\restriction}_k < \sum_{i=1}^{n} 2^{-|a_i|}$が成り立つ。この不等式から、任意の$i > n$に対して$|a_i| > n$となることがわかる。従って、$\{a_1, \ldots, a_n\}$は、$R$の定義域の元のうち、長さが$k$以下のものを全て含んでいる。これゆえ、$\{R(a_1), \ldots, R(a_n)\}$にない$x$を選べば、$k < K_R(x)$である。このようにして、$k < K_R(f(\Omega_R{\restriction}_k))$が成り立つ。

補題より、

$$k < K_R(f(\Omega_R{\restriction}_k)) \leq K(f(\Omega_R{\restriction}_k)) + O(1) \leq K(\Omega_R{\restriction}_k) + O(1).$$

□

測度論に基づくランダムネスの定義

準備: 半決定可能集合と帰納的枚挙可能集合

定義 [半決定プログラム] kを1以上の自然数とする。集合$\alpha \subset \mathbb{N}^k$と$k$入力プログラム$\mathcal{Q}$に対し、次が成り立つとき、$\mathcal{Q}$は$\alpha$を半決定する、という。

> $\mathcal{Q}$を入力値$\vec{x}$で実行すると、$\vec{x} \in \alpha$の場合は何らかの値を返して実行が終了し、$\vec{x} \notin \alpha$の場合は、実行が永久に終了しない。

□

定義 [半決定可能集合]

kを1以上の任意の自然数とする。集合$\alpha \subset \mathbb{N}^k$に対し、$\alpha$を半決定する$k$入力プログラムが存在するとき、$\alpha$は半決定可能 (semi-decidable)である、という。 □

準備: 半決定可能集合と帰納的枚挙可能集合

`print`文とは次の形をした文のことであり、これを実行すると数式の値が計算されてそれが出力される。

`printf(` 数式 `);`

定義 [枚挙プログラム] 自然数の集合αとプログラム$\mathcal{P}$に対し「$\mathcal{P}$はαを枚挙する」とは、次の２条件を満たすこととして定義する。

(i) $\mathcal{P}$には入力変数と`return`文がなく、`printf`文はいくつあってもよい。$\mathcal{P}$中で呼び出して使用する他の関数プログラムはの書式は`printf`文を含まない通常どうりのものであるとする。

(ii) $\alpha = \{n \in \mathbb{N}^k \mid \mathcal{P}$を実行するといつかは$n$が出力される$\}$. つまり、$\mathcal{P}$の実行によって出力される自然数全体の集合が$\alpha$に等しい。 □

定義 [帰納的枚挙可能集合]

(i) 自然数の集合αに対し、αを枚挙するプログラムが存在するとき、αは帰納的枚挙可能 (recursively enumerable)である、という。

(ii) kを自然数とする。集合$\alpha \subset \mathbb{N}^k$に対し、集合$\{\langle \vec{x} \rangle \mid \vec{x} \in \alpha\} \subset \mathbb{N}$が枚挙可能のとき、$\alpha$は帰納的枚挙可能である、という。 □

準備: 半決定可能集合と帰納的枚挙可能集合

定理 [半決定可能性と帰納的枚挙可性の同値性] k を１以上の任意の自然数とする。集合 $\alpha \subset \mathbb{N}^k$ に関する次の３条件は、すべて同値である。

(i) α は半決定可能である。

(ii) α は帰納的枚挙可能である。

(iii) α を定義域とする k 変数計算可能部分関数が存在する。 □

Martin-Löfランダムネス

基本的な考え方

$2^{\mathbb{N}}$のどのような構成的零集合にも属さない無限２進列が、ランダムな列である。 □

これを正当化する思考実験

1. 偏りのないコインを無限回投げて生成した具体的な無限２進列が“ ランダム ”である。
2. $A_1, A_2, A_3, \ldots\ldots$ をアルゴリズムによって“ 機械的に ”生成した$2^{\mathbb{N}}$の部分集合列とし、$\lambda(A_n) \leq 2^{-n}$が成り立っているとする（λはLebesgue測度）。
3. 前提: Xを与えられた無限２進列とする。$X \in \bigcap_n A_n$が成り立つとする。
4. もし仮にXが“ ランダム ”であるならば、Xは偏りのないコインを無限回投げて生成したものである。一方、各A_nは、アルゴリズムという機械が指定する事象であるが、それらが生起する確率は$\lambda(A_n) \to 0$となる。従って、確率がいくらでも小さい人工的な事象が生じたことになる。これはおかしい。仮定が間違っている。
5. 結論: Xはランダムでない。 □

Martin-Löfランダムネス

定義 有限2進列の集合からなる列$\{G_n\}_{n\in\mathbb{N}}$が一様に帰納的枚挙可能(uniformly recursively enumerable)であるとは、集合$\{(\sigma, n) \mid \sigma \in G_n\} \subset \mathbb{N}^2$が帰納的枚挙可能であることをいう。 □

定義 [Martin-Löf 1966]

(i) Martin-Löfテストとは、有限2進列の集合列$\{G_n\}_{n\in\mathbb{N}}$であって、一様に帰納的枚挙可能かつ、次を満たすもののことをいう。

$$\forall n \in \mathbb{N} \quad \lambda\left([G_n]^{\prec}\right) \leq 2^{-n}.$$

ここで、$[G_n]^{\prec} = \{X \in 2^{\mathbb{N}} \mid \text{ある自然数} n \text{が存在して} X\restriction_n \in G_n\}$である。

(ii) 無限2進列$X \in 2^{\mathbb{N}}$がMartin-Löfランダムであるとは、任意のMartin-Löfテスト$\{G_n\}_{n\in\mathbb{N}}$に対し、次が成り立つことを言う。

$$X \notin \bigcap_{n=0}^{\infty} [G_n]^{\prec}.$$ □

Martin-Löfテストの例

定理 M をprefix-free machineとする。各自然数nに対し、

$$R_n^M = \{x \in \{0,1\}^* \mid K_M(x) \leq |x| - n\}$$

とおく。このとき、有限2進列の集合列$\{R_n^M\}_{n\in\mathbb{N}}$はMartin-Löfテストである。

証明)

はじめに、各nに対し、次が成り立つ。

$$\lambda\left(\left[R_n^M\right]^{\prec}\right) \leq \sum_{x\in R_n^M} 2^{-|x|} \leq \sum_{x\in R_n^M} 2^{-K_M(x)-n} \leq \sum_{x\in\{0,1\}^*} 2^{-K_M(x)-n}$$
$$= 2^{-n} \sum_{x\in\{0,1\}^*} 2^{-K_M(x)} \leq 2^{-n}.$$

Martin-Löfテストの例

証明（続き））

ある自然数pが存在し$R(s) = \mathsf{comp}(p, \langle s \rangle)$が成り立つ。集合$\{(x, n) \mid x \in R_n^M\} \subset \mathbb{N}^2$は、次の２入力関数プログラムで半決定されるので、$\{R_n^M\}_{n \in \mathbb{N}}$は帰納的枚挙可能である。

```
R(x,n){
int s, t, c;
  c = 1;
  while (0 == 0) {
    s = left(c);
    t = right(c);
    if (halt_before(p,pair(s,0),t) == 1) {
      if (comp(p,pair(s,0)) == x) && (strlen(s) <= strlen{x} - n))
        return(1);
    }
    c ++;
  }
}
```

□

2つのランダムネスの同値性

定理 [Schnorr 1973] X を任意の無限2進列とする。次の2つの条件は互いに同値である。

(i) X は1-ランダム。

(ii) X はMartin-Löfランダムである。 □

(ii) ⇒ (i) の証明)
対偶を示す。X が1-ランダムでないと仮定する。このとき、任意の n に対して、ある k が存在し、

$$K(X\restriction_k) + n < k$$

が成り立つ。さて、Martin-Löfテスト $\{R_n^U\}_{n\in\mathbb{N}}$ を考えると、任意の n に対して、ある k が存在し、

$$X\restriction_k \in R_n^U$$

が成り立つが、これは $X \in \bigcap_{n=1}^{\infty} R_n^U$ を意味する。従って、X はMartin-Löfランダムではない。 □

Kraft-Chaitinの定理

定理 [Chaitin 1975] $f\colon \mathbb{N} \to \mathbb{N}$を計算可能全域関数とし、$\sum_{n=0}^{\infty} 2^{-f(n)} \leq 1$を満たしているものとする。このとき、ある計算可能な全域関数$g\colon \mathbb{N} \to \{0,1\}^*$が存在して、次を満たす。

(i) gは単射。

(ii) $g(\mathbb{N}) := \{g(n) \mid n \in \mathbb{N}\}$はprefix-free集合。

(iii) 任意の自然数nに対して、$|g(n)| = f(n)$. □

2つのランダムネスの同値性

定理 [Schnorr 1973, 再掲] X を任意の無限2進列とする。次の2つの条件は互いに同値である。

(i) X は1-ランダム。

(ii) X はMartin-Löfランダムである。 □

(i) ⇒ (ii) の証明)

対偶を示す。X がMartin-Löfランダムでないと仮定する。このとき、あるMartin-Löfテスト $\{G_n\}_{n\in\mathbb{N}}$ が存在して、任意の n に対して、ある m が存在して、

$$X\restriction_m \in G_n$$

が成り立つ。$\{G_n\}$ は一様に帰納的枚挙可能なので、$F_n = G_{2n}$ とおくと、$\{F_n\}$ も一様に帰納的枚挙可能である。従って、1変数計算可能全域関数 n, s が存在し、集合 $\{(l,t) \mid t \in F_l\}$ の元は、過不足なく、

$$(n(0), s(0)), (n(1), s(1)), (n(2), s(2)), \ldots$$

と列挙される。即ち、$\{(n(i), s(i)) \mid i \in \mathbb{N}\} = \{(l,t) \mid t \in F_l\}$ である。

2つのランダムネスの同値性

(i) ⇒ (ii) の証明）（続き）

さて、1変数計算可能全域関数 $f\colon \mathbb{N} \to \mathbb{N}$ を $f(k) = |s(k)| - n(k)$ で定義する。このとき、

$$\begin{aligned}
\sum_{k=0}^{\infty} 2^{-f(k)} &= \sum_{k=0}^{\infty} 2^{-|s(k)|+n(k)} = \sum_{s\in F_n} 2^{-|s|+n} = \sum_{s\in G_{2n}} 2^{-|s|+n} \\
&= \sum_{n=1}^{\infty} \sum_{s\in G_{2n}} 2^{-|s|+n} = \sum_{n=1}^{\infty} 2^n \sum_{s\in G_{2n}} 2^{-|s|} \\
&\leq \sum_{n=1}^{\infty} 2^n 2^{-2n} \qquad (\{G_n\} \text{はMartin-Löfテストなので}) \\
&= \sum_{n=1}^{\infty} 2^{-n} = 1
\end{aligned}$$

となる。従って、Kraft-Chaitinの定理より、ある計算可能な全域関数 $g\colon \mathbb{N} \to \{0,1\}^*$ が存在して、次を満たす。

(i) g は単射。

(ii) $g(\mathbb{N}) := \{g(n) \mid n \in \mathbb{N}\}$ はprefix-free集合。

(iii) 任意の n に対して、$|g(k)| = f(k) = |s(k)| - n(k)$. □

2つのランダムネスの同値性

(i) ⇒ (ii) の証明）（続き）

次のような関数プログラムMを考える。

```
M(t){
int k;
  k = 0;
  while (g(k) == t) {
    k ++;
  }
  return(s(k));
  }
}
```

このとき、M の定義域は $g(\mathbb{N})$ であり、M はprefix-free machineである。また、g は単射であるから、任意の k に対し $M(g(k)) = s(k)$ であることもわかる。従って、

$$K_M(s(k)) \leq |g(k)| = f(k) = |s(k)| - n(k)$$

を得る。

　さて、任意の n に対して、ある m が存在して、$X\restriction_m \in G_{2n}$ であったが、これは $X\restriction_m \in F_n$ を意味し、ある k が存在し、$(n, X\restriction_m) = (n(k), s(k))$ となる。従って、

$$K(X\restriction_m) \leq K_M(X\restriction_m) + O(1) \leq |X\restriction_m| - n + O(1).$$

が成り立つ。これゆえ、X は1-ランダムではない。 □

万能Martin-Löfテスト

定義 [万能Martin-Löfテスト] $\{U_n\}_{n\in\mathbb{N}}$をMartin-Löfテストとする。$\{U_n\}_{n\in\mathbb{N}}$が万能 (universal)であるとは、任意のMartin-Löfテスト$\{G_n\}_{n\in\mathbb{N}}$に対し、次が成り立つことである。

$$\bigcap_{n=1}^{\infty} [G_n]^{\prec} \subset \bigcap_{n=1}^{\infty} [U_n]^{\prec}.$$

□

定理 Uをoptimal prefix-free machineとする。このとき$\{R_n^U\}_{n\in\mathbb{N}}$は万能Martin-Löfテストである。ここで$R_n^U = \{x \in \{0,1\}^* \mid K_U(x) \leq |x| - n\}$.

証明)

$\{G_n\}_{n\in\mathbb{N}}$を任意のMartin-Löfテストとする。任意の無限2進列Xに対し、$X \in \bigcap_{n=1}^{\infty} [G_n]^{\prec}$と仮定すると、$X$はMartin-Löfランダムでないので、1-ランダムでもない。これは、任意の任意のnに対して、あるkが存在し、

$$K(X\restriction_k) < k - n$$

となることを意味し、$X \in \bigcap_{n=1}^{\infty} \left[R_n^U\right]^{\prec}$が成り立つ。従って、$\{R_n^U\}_{n\in\mathbb{N}}$は万能Martin-Löfテストである。 □

参考文献

[1] G. J. Chaitin, "A theory of program size formally identical to information theory," *J. Assoc. Comput. Mach.*, vol.22, pp.329–340, 1975.

[2] 鹿島亮,『C言語による計算の理論』, Computer Science Library 4, サイエンス社, 2008.

[3] A. Nies, *Computability and Randomness*. Oxford University Press, Inc., New York, 2009.

[4] R. G. Downey and D. R. Hirschfeldt, *Algorithmic Randomness and Complexity*. Springer-Verlag, New York, 2010.

[5] 只木孝太郎,『アルゴリズム的情報理論入門』, 集中講義ノート, 大学院数学レクチャーノートシリーズ TML26, 田中一之監修, 東北大学大学院理学研究科, 2011. (`http://www2.odn.ne.jp/tadaki/texts.html`からダウンロード可能)