計算理論　第 4 回
コード化

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2020 年 10 月 29 日

最終更新：2020 年 10 月 28 日　14:22

概要

この講義の主題

計算理論 (Theory of Computation)

- 計算可能性理論　(Computability Theory)
- 計算複雑性理論 (計算量理論)　(Complexity Theory)

講義の進め方

- 前半：計算可能性理論 (担当：岡本)
- 後半：計算複雑性理論 (担当：垂井先生)

スケジュール 前半 (予定)

1. 計算とは何か？　(10/1)
2. 計算モデル　(10/8)
3. チャーチ・チューリングの定立　(10/15)
- ⋆ 休み (体育祭)　(10/22)
4. コード化　(10/29)
5. 計算可能性　(11/5)
6. 停止性問題　(11/12)
7. 再帰定理　(11/19)
8. 前半のまとめ　(11/26)

注意：予定の変更もありうる

いままでの話 と 今日の話

いままでの話

- 自然数 k 個を変数とする部分関数の計算を考える
- 計算モデルとして，簡単なプログラミング言語を考える
  - WHILE プログラム，LOOP プログラム，GOTO プログラム

疑問：自然数しか入力できないと，計算として不自由 (強い制限) では？

今日の話

- 自然数以外のデータを 自然数として表現する
- 特に，プログラムを 自然数として表現する

$\rightsquigarrow$ 「プログラムを入力とするプログラム」を考えられるようになる

考える順番：対 (ペア) → リスト → プログラム

目次

コード化 (符号化)

集合 $\mathcal{A}$

コード化とは？ (直観的な説明)

$\mathcal{A}$ のコード化とは，各 $a \in \mathcal{A}$ に対して，
異なる自然数 $\mathrm{enc}(a)$ を割り当てる関数 $\mathrm{enc}\colon \mathcal{A} \to \mathbb{N}$

異なる自然数を割り当てるので，enc は単射

復習：単射

関数 $f\colon A \to B$ が単射であるとは，次の条件を満たすこと

任意の $a, a' \in A$ に対して，$f(a) = f(a')$ ならば，$a = a'$

$\mathrm{enc}(a)$ を a のコード (符号) と呼ぶ

デコード化 (復号化)

集合 $\mathcal{A}$，コード化 $\mathrm{enc}\colon \mathcal{A} \to \mathbb{N}$

デコード化とは？

enc に対するデコード化とは，次の部分関数 $\mathrm{dec}\colon \mathbb{N} \nrightarrow \mathcal{A}$

$$\begin{array}{lcl} n = \mathrm{enc}(a) \text{ となる } a \text{ が存在する} & \Rightarrow & \mathrm{dec}(n) = a \\ n = \mathrm{enc}(a) \text{ となる } a \text{ が存在しない} & \Rightarrow & \mathrm{dec}(n) \uparrow \end{array}$$

注

- $n = \mathrm{enc}(a)$ となる a が存在するとき，そのような a は 1 つしかない
- 任意の $a \in \mathcal{A}$ に対して，$\mathrm{dec}(\mathrm{enc}(a)) = a$ となる
- dec は enc に対して決まる ($\mathrm{dec}_{\mathrm{enc}}$ と書く方が正確かもしれない)

目次

自然数の対

定義：対 (つい，ペア)

2 つの (同じかもしれない) 自然数 $a, b \in \mathbb{N}$ に対して，
それらを組にしたもの (a, b) を a, b の対と呼ぶ

例：$(1, 3)$，$(2, 2)$，$(5, 0)$ など

自然数の対の集合

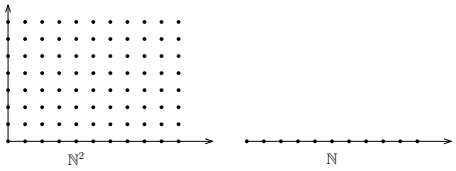
$\mathbb{N}^2$ = 自然数の対をすべて集めた集合

自然数の対のコード化

考えたいこと

自然数の対のコード化

つまり，単射 $\mathrm{enc}\colon \mathbb{N}^2 \to \mathbb{N}$



$\mathbb{N}^2$ の方が $\mathbb{N}$ より「圧倒的に大きい」のに，単射はあるのか？

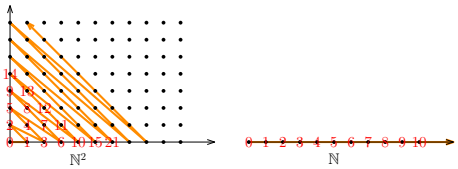
カントールの対関数

様々な単射 $\mathbb{N}^2 \to \mathbb{N}$ が知られているが，
この講義ではカントールの対関数 (ついかんすう) を用いる

定義：カントールの対関数

次の関数 $\pi\colon \mathbb{N}^2 \to \mathbb{N}$ をカントールの対関数と呼ぶ

$$\pi(a, b) = \frac{1}{2}(a+b)(a+b+1) + b$$



$\pi(0,0) = 0$, $\pi(1,0) = 1$, $\pi(0,1) = 2$, $\pi(2,0) = 3$, $\pi(1,1) = 4$, $\ldots$

カントールの対関数：単射性

性質：カントールの対関数の単射性

カントールの対関数 $\pi\colon \mathbb{N}^2 \to \mathbb{N}$ は単射である，すなわち，

$$\pi(a_1, b_1) = \pi(a_2, b_2) \quad \Rightarrow \quad (a_1, b_1) = (a_2, b_2)$$

証明：演習問題

- ヒント：$\pi(a_1, b_1) = \pi(a_2, b_2)$ を仮定して，
  まず $a_1 + b_1 = a_2 + b_2$ となることを証明する

注：π は全射でもあるので，π は全単射である

カントールの対関数：WHILE 計算可能性

性質：カントールの対関数の WHILE 計算可能性

カントールの対関数 $\pi\colon \mathbb{N}^2 \to \mathbb{N}$ は WHILE 計算可能

証明：次のプログラムが π を計算する

$x_3 := \mathtt{add}(x_1, x_2);$
$x_4 := 1;$
$x_5 := \mathtt{add}(x_3, x_4);$
$x_6 := \mathtt{mult}(x_3, x_5);$
$x_7 := \mathtt{half}(x_6);$
$x_0 := \mathtt{add}(x_7, x_2)$

これは糖衣構文を含んでいるが，それらを書き換えると
WHILE プログラムが得られる □

今後，π を計算する WHILE プログラムを pi と書くことにする

カントールの対関数：デコード化 (1)

カントールの対関数のデコード化

$\frac{1}{2}(a+b)(a+b+1) + b$ から a と b を復元するには？

$x_1 = \frac{1}{2}(a+b)(a+b+1) + b$ とする

x_1 から a を計算するプログラム fst

$x_2 := x_1;\ x_2 := x_2 + 1;\ x_3 := x_1;\ x_3 := x_3 + 1;$
LOOP x_2 DO
  $x_6 := 0;$
  LOOP x_3 DO
    $x_4 := \mathtt{pi}(x_5, x_6);$
    IF $x_1 = x_4$ THEN $x_0 := x_5$ END;
    $x_6 := x_6 + 1$
  END;
  $x_5 := x_5 + 1$
END

カントールの対関数：デコード化 (2)

カントールの対関数のデコード化

$\frac{1}{2}(a+b)(a+b+1) + b$ から a と b を復元するには？

$x_1 = \frac{1}{2}(a+b)(a+b+1) + b$ とする

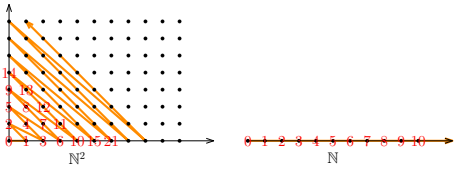
x_1 から b を計算するプログラム snd

$x_2 := x_1;\ x_2 := x_2 + 1;\ x_3 := x_1;\ x_3 := x_3 + 1;$
LOOP x_2 DO
  $x_6 := 0;$
  LOOP x_3 DO
    $x_4 := \mathtt{pi}(x_5, x_6);$
    IF $x_1 = x_4$ THEN $x_0 := x_6$ END;
    $x_6 := x_6 + 1$
  END;
  $x_5 := x_5 + 1$
END

ペアのコード化・デコード化：まとめ

カントールの対関数 $\pi\colon \mathbb{N}^2 \to \mathbb{N}$

$$\pi(a, b) = \frac{1}{2}(a+b)(a+b+1) + b$$



コード化，デコード化は WHILE 計算可能

- コード化関数 pi　　$((a, b) \mapsto \pi(a, b))$
- デコード化関数 fst　　$(\pi(a, b) \mapsto a)$
- デコード化関数 snd　　$(\pi(a, b) \mapsto b)$

目次

リスト

自然数のリストとは？

ある自然数 n に対して，n 個の自然数を並べたもの

$$(a_1, a_2, \ldots, a_n)$$

n をこのリストの長さと呼ぶ

自然数のリストの集合

- $\mathbb{N}^n$ = 自然数のリストで長さが n のものをすべて集めた集合
- $\mathbb{N}^* = \bigcup_{n \in \mathbb{N}} \mathbb{N}^n$ (自然数のリストをすべて集めた集合)

注：長さ 0 のリスト ( ) も考えている

リストのコード化 (1)

長さによる帰納法で，リストのコード化を行う

- 長さ $n = 0$ のとき (リストは ( ))

$$\mathsf{enc}(( )) = 0$$

- 長さ $n = 1$ のとき (リストは (a_1))

$$\begin{aligned}\mathsf{enc}((a_1)) &= \pi(a_1 + 1, \mathsf{enc}(( ))) \\ &= \pi(a_1 + 1, 0)\end{aligned}$$

- 長さ $n = 2$ のとき (リストは (a_1, a_2))

$$\begin{aligned}\mathsf{enc}((a_1, a_2)) &= \pi(a_1 + 1, \mathsf{enc}((a_2))) \\ &= \pi(a_1 + 1, \pi(a_2 + 1, 0))\end{aligned}$$

- ……

リストのコード化 (2)

- 長さ $n = 3$ のとき (リストは (a_1, a_2, a_3))

$$\begin{aligned}\mathsf{enc}((a_1, a_2, a_3)) &= \pi(a_1 + 1, \mathsf{enc}((a_2, a_3))) \\ &= \pi(a_1 + 1, \pi(a_2 + 1, \pi(a_3 + 1, 0)))\end{aligned}$$

- 長さ n のとき (リストは $(a_1, a_2, \ldots, a_n)$)

$$\mathsf{enc}((a_1, a_2, \ldots, a_n)) = \pi(a_1 + 1, \mathsf{enc}((a_2, \ldots, a_n)))$$

例

$$\begin{aligned}\mathsf{enc}((3, 0, 2)) &= \pi(3 + 1, \mathsf{enc}((0, 2))) = \pi(4, \pi(0 + 1, \mathsf{enc}((2)))) \\ &= \pi(4, \pi(1, \pi(2 + 1, \mathsf{enc}(( ))))) = \pi(4, \pi(1, \pi(3, 0))) \\ &= \pi(4, \pi(1, 6)) = \pi(4, 34) = 775\end{aligned}$$

リストのコード化：単射性 (1)

補題

この $\mathsf{enc}\colon \mathbb{N}^* \to \mathbb{N}$ は単射である

まず，$e = \mathsf{enc}((a_1, a_2, \ldots, a_n))$ から n が復元できることを示す

次のアルゴリズムを考える

1. $n := 0$ とする
2. $e = 0$ ならば，n を出力
3. $e \neq 0$ ならば，次を実行
   1. $e := \mathsf{snd}(e)$
   2. $n := n + 1$
   3. ステップ 2 に戻る

これで正しく n が復元できる

証明のアイディア：
コード化において，π の引数として 0 が現れるのは，長さ 0 のときのみ

リストのコード化：単射性 (2)

$e = \mathsf{enc}((a_1, a_2, \ldots, a_n))$ から n が復元できたので，
あとは，$a_1, a_2, \ldots$ を順に復元する

e, n から $a_1, \ldots, a_n$ を復元するアルゴリズム

1. $i := 1$
2. 次を n 回繰り返す
   1. $a_i := \mathsf{fst}(e) - 1$
   2. $e := \mathsf{snd}(e)$
   3. $i := i + 1$

これで正しく $a_1, a_2, \ldots, a_n$ が復元できる

目次

プログラムのコード化

プログラムのコード化

リストのコード化を用いることで，プログラムのコード化も行える

- WHILE プログラムのコード化を考えるのは ちょっと厄介
  (WHILE プログラムは再帰的に定義されているため)
- 代わりに，GOTO プログラムのコード化を考える

WHILE プログラム P をコード化しようと思ったら，
P を模倣する GOTO プログラムをコード化すればよい

復習 — GOTO プログラム：文法 (1)

GOTO プログラムの文法 (構文論)

GOTO プログラムは次の形をしている

$$\begin{array}{l} L_1\colon\ S_1; \\ L_2\colon\ S_2; \\ \quad\vdots \\ L_n\colon\ S_n \end{array}$$

- $L_1, L_2, \ldots, L_n$ はラベル
- $S_1, S_2, \ldots, S_n$ は文 (次のページで定義される)

復習 — GOTO プログラム：文法 (2)

GOTO プログラムの文法 (構文論) 続き

GOTO プログラムの文は次のいずれか

1. 変数 x に対して，「$x := x + 1$」
2. 変数 x に対して，「$x := x - 1$」
3. ラベル L に対して，「GOTO L」
4. 変数 x とラベル L に対して，「IF $x = 0$ THEN GOTO L」
5. 「HALT」

$$\begin{array}{l} L_1\colon\ \texttt{IF}\ x_1 = 0\ \texttt{THEN GOTO}\ L_5; \\ L_2\colon\ x_0 := x_0 + 1; \\ L_3\colon\ x_1 := x_1 - 1; \\ L_4\colon\ \texttt{GOTO}\ L_1; \\ L_5\colon\ \texttt{IF}\ x_2 = 0\ \texttt{THEN GOTO}\ L_9; \\ L_6\colon\ x_0 := x_0 + 1; \\ L_7\colon\ x_2 := x_2 - 1; \\ L_8\colon\ \texttt{GOTO}\ L_5; \\ L_9\colon\ \texttt{HALT} \end{array}$$

GOTO プログラムのコード化 (1) 基本アイディア

$$\begin{array}{l} L_1\colon\ S_1; \\ L_2\colon\ S_2; \\ \quad\vdots \\ L_n\colon\ S_n \end{array}$$

GOTO プログラムをどのようにコード化するか？

1. 各文 $S_1, S_2, \ldots, S_n$ をコード化する
   $\rightsquigarrow \mathsf{enc}(S_1), \mathsf{enc}(S_2), \ldots, \mathsf{enc}(S_n)$
2. それらと k，m を並べてリストを作る
   - k = 入力変数の数
   - m = 使用する変数の添え字の最大値

   $\rightsquigarrow (k, m, \mathsf{enc}(S_1), \mathsf{enc}(S_2), \ldots, \mathsf{enc}(S_n))$ (←自然数のリスト)
3. そのリストを (先ほどの方法で) コード化する
   $\rightsquigarrow \mathsf{enc}((k, m, \mathsf{enc}(S_1), \mathsf{enc}(S_2), \ldots, \mathsf{enc}(S_n)))$

GOTO プログラムのコード化 (2) 文のコード化

文	コード
$x_i := x_i + 1$	$\mathsf{enc}((1, i))$
$x_i := x_i - 1$	$\mathsf{enc}((2, i))$
GOTO L_j	$\mathsf{enc}((3, j))$
IF $x_i = 0$ THEN GOTO L_j	$\mathsf{enc}((4, i, j))$
HALT	$\mathsf{enc}((5))$

GOTO プログラムのコード化：例 (1)

関数 add: $\mathbb{N}^2 \to \mathbb{N}$ を計算する次の GOTO プログラムをコード化する

$$\begin{array}{l} L_1\colon\ \texttt{IF}\ x_1 = 0\ \texttt{THEN GOTO}\ L_5; \\ L_2\colon\ x_0 := x_0 + 1; \\ L_3\colon\ x_1 := x_1 - 1; \\ L_4\colon\ \texttt{GOTO}\ L_1; \\ L_5\colon\ \texttt{IF}\ x_2 = 0\ \texttt{THEN GOTO}\ L_9; \\ L_6\colon\ x_0 := x_0 + 1; \\ L_7\colon\ x_2 := x_2 - 1; \\ L_8\colon\ \texttt{GOTO}\ L_5; \\ L_9\colon\ \texttt{HALT} \end{array}$$

このプログラムにおいて，

- 入力変数の数 $k = 2$
- 使用する変数の添え字の最大値 $m = 2$

GOTO プログラムのコード化：例 (2)

各文のコード化

$$\begin{array}{l} L_1\colon\ \texttt{IF}\ x_1 = 0\ \texttt{THEN GOTO}\ L_5; \\ L_2\colon\ x_0 := x_0 + 1; \\ L_3\colon\ x_1 := x_1 - 1; \\ L_4\colon\ \texttt{GOTO}\ L_1; \\ L_5\colon\ \texttt{IF}\ x_2 = 0\ \texttt{THEN GOTO}\ L_9; \\ L_6\colon\ x_0 := x_0 + 1; \\ L_7\colon\ x_2 := x_2 - 1; \\ L_8\colon\ \texttt{GOTO}\ L_5; \\ L_9\colon\ \texttt{HALT} \end{array}$$

$\mathsf{enc}((4,1,5)) = \pi(5, \pi(2, \pi(6,0))) = 46050$
$\mathsf{enc}((1,0)) = \pi(2, \pi(1,0)) = 7$
$\mathsf{enc}((2,1)) = \pi(3, \pi(2,0)) = 24$
$\mathsf{enc}((3,1)) = \pi(4, \pi(2,0)) = 31$
$\mathsf{enc}((4,2,9)) = \pi(5, \pi(3, \pi(10,0))) = 1570872$
$\mathsf{enc}((1,0)) = \pi(2, \pi(1,0)) = 7$
$\mathsf{enc}((2,2)) = \pi(3, \pi(3,0)) = 51$
$\mathsf{enc}((3,5)) = \pi(4, \pi(6,0)) = 346$
$\mathsf{enc}((5)) = \pi(6,0) = 21$

文	コード
$x_i := x_i + 1$	$\mathsf{enc}((1, i))$
$x_i := x_i - 1$	$\mathsf{enc}((2, i))$
GOTO L_j	$\mathsf{enc}((3, j))$
IF $x_i = 0$ THEN GOTO L_j	$\mathsf{enc}((4, i, j))$
HALT	$\mathsf{enc}((5))$

GOTO プログラムのコード化：例 (3)

k, m と各文のコードを並べてリストを作り，コード化する

$$\begin{aligned} &\mathsf{enc}((2, 2, 46050, 7, 24, 31, 1570872, 7, 51, 346)) \\ &= \pi(3, \pi(3, \pi(46051, \pi(8, \pi(25, \pi(32, \pi(1570873, \pi(8, \pi(52, \pi(347, 0)))))))))) \end{aligned}$$

これを計算した結果は次のページの通り

GOTO プログラムのコード化：例 (4)

151536467753926064823310150718467326468895050732596919135802527354672640284
406667224493240577231180334762796972732977536390877035024430040030458257199
442675021109712531259139724889109553030599431021903229989939683346299584230
939849320724370424219271691313009156521549001673404496254234337397670804106
765338495795965531477730742692564423978556570444671227178432924461577079085
350286302246090259399762570012407048037342309672387989072473804721472262297
943185204331087112206342506687583454351674270116480014960366000545023480571
214484757217041120341563945233678504758485760800216994204836876399547686218
807318516451797125525744252058683805991559676791813418181842482430666565617
007617509710343530737297496749704985862980148285559583289880741230932051714 1
097272103502874434020098071111788632410867016726584732877145779270674338603
466690768551857041116038498778637583927147511449262135268197218312001445337
843154327871797226335221750672895493163327659584168844282240050392074960921
204097904936862960126577732797317665202639039959944036112085332327647334602
418445588658888899883255582402967452426365272959518007307952663568813836614 2
932379968139949526637130362431703199098293345062526688175705496343948929776
746647857662934031875385418111883677284401678172151995375618981405636689906 2
706594101775366169606165875349763361295516135869032859623194940557057341321
635311778318931167851251109143206714043343378417424190843931046774442295845
354458264970058261627256321496813534294015427447388419272139012548149196411
664986821802351097443198068962993542830154741674746165295372251951331211988
045319154264489754669404394741159050786877680487895388814467035272346419772
684744044793198569699776336206387175083609051352270381759338985785360121703
275692376615227817320789471307463323961817479054959452057011899591668412085
786337274851649750954931655342690056507066676230719309362213478299961998 498
491312150387579772847455207959699625403489188276904585683629574510754825595
331178611670732194484151018915454320350042343921727988122190897596243083863
340358201003583650807786522986913395470693564281510306050411290889514610289
118420075103173540572613357447566990143360964835299742554151746143619236102
452830352730370087557371277795901026812014110206093800180008909180433951735
240382010911679697179224308797854445218483557

目次

リストを処理するプログラム (1)

次回，リストを処理するプログラムが必要となるので，いくつか導入する

次の関数 isList: $\mathbb{N} \to \mathbb{N}$ を考える

$$\mathtt{isList}(x_1) = \begin{cases} 1 & (x_1 \text{ があるリストのコードであるとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

関数 isList は WHILE 計算可能である

リストのコードであることを判定する関数 isList

```
x_0 := 1;
WHILE x_1 ≠ 0 DO
  x_2 := fst(x_1);
  IF x_2 = 0 THEN x_0 := 0 END;
  x_1 := snd(x_1);
END
```

備忘録：

$$\mathsf{enc}((a_1, \ldots, a_n)) = \begin{cases} 0 & (n = 0 \text{ のとき}) \\ \pi(a_1 + 1, \mathsf{enc}((a_2, \ldots, a_{n-1}))) & (n \geq 1 \text{ のとき}). \end{cases}$$

リストを処理するプログラム (2)：リストの長さを出力する部分関数 len

次の部分関数 len: $\mathbb{N} \rightharpoonup \mathbb{N}$ を考える

$$\mathtt{len}(x_1) = \begin{cases} n & (x_1 \text{ が長さ } n \text{ のリストのコードであるとき}) \\ \text{定義されない} & (\text{そうではないとき}) \end{cases}$$

部分関数 len は WHILE 計算可能である

```
x_2 := isList(x_1);
IF x_2 ≠ 0 THEN
  WHILE x_1 ≠ 0 DO
    x_1 := snd(x_1); x_0 := x_0 + 1
  END
ELSE
  infloop(x_1)
END
```


リストを処理するプログラム (3)：リストの i 番目の要素を出力する elem

次の部分関数 elem: $\mathbb{N}^2 \rightharpoonup \mathbb{N}$ を考える

$$\mathtt{elem}(e, i) = \begin{cases} a_i & (e \text{ がリスト } (a_1, \ldots, a_n) \text{ のコードであり，} 1 \leq i \leq n \text{ のとき}) \\ \text{定義されない} & (\text{そうではないとき}) \end{cases}$$

部分関数 elem は WHILE 計算可能 ($x_1 = e$，$x_2 = i$ とする)

```
x_3 := len(x_1); x_3 := x_3 + 1; x_4 := sub(x_3, x_2);
IF x_4 = 0 THEN infloop(x_1) END; IF x_2 = 0 THEN infloop(x_1) END;
x_2 := x_2 − 1;
WHILE x_2 ≠ 0 DO
  x_1 := snd(x_1); x_2 := x_2 − 1
END;
x_0 := fst(x_1); x_0 := x_0 − 1
```


リストを処理するプログラム (4)：リストの i 番目の要素を変更する replace

次の部分関数 replace: $\mathbb{N}^2 \rightharpoonup \mathbb{N}$ を考える

$$\mathtt{replace}(e, i, x) = \begin{cases} \mathsf{enc}((a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n)) \\ \quad (e \text{ がリスト } (a_1, \ldots, a_n) \text{ のコードであり，} \\ \quad 1 \leq i \leq n \text{ であるとき}) \\ \text{定義されない} \\ \quad (\text{そうではないとき}) \end{cases}$$

部分関数 replace は WHILE 計算可能 (演習問題)

リストを処理するプログラム (5)：GOTO プログラムのコードであることを判定する

次の部分関数 isProg: $\mathbb{N} \rightharpoonup \mathbb{N}$ を考える

$$\mathtt{isProg}(x_1) = \begin{cases} 1 & (x_1 \text{ が ある GOTO プログラムのコードであるとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

isProg は WHILE 計算可能である (演習問題)

目次

今日のまとめ と 次回の予告

今日のまとめ

コード化 (自然数以外のデータを自然数として表す)

- 対 (ペア)： カントールの対関数を考える
- リスト： 再帰的なペアと見なす
- プログラム： GOTO プログラムをリストと見なす
- 特に，プログラムを 自然数として表現する

注：講義で紹介した方法以外にも，コード化の方法は存在する
(考えてみよう)

次回の予告：この講義のハイライト (1)

- WHILE 計算不可能な部分関数が存在することの証明 (対角線論法)
- 万能プログラム (インタプリタ) の設計