東京女子大学 現代の数学とその応用(2003 年度) 資料

非常勤講師 永島(ながしま) 孝(たかし)

E-mail　nagasima-t@mta.biglobe.ne.jp
Web site　http://www.twcu.ac.jp/~nagasima/
個人Web site　http://www2s.biglobe.ne.jp/~hotori/

1　数とその表記法, 十進法, 二進法など

ものを数えるとき, 十個ずつまとめていく数え方が**十進法**の基本である. 十個まとめたものをさらに十個まとめて百個, それをまた十個まとめて千個などとして, それらのまとまりがいくつあるかを考える. まとめるときの単位として慣習的に十が選ばれているのは, 指の数に由来するというのが定説である. 十個ずつまとめて数えるときの十を**底**(てい)[1]と呼ぶ. 一般に, 1 より大きい整数 p に対して, p を底にする数え方を p 進法と呼ぶ.

古代文明での数え方を見ると十進法が多く, エジプト, インド, 中国などが十進法をもちいていて, それぞれ十進法に基く数字を創り出している. それ以外にはバビロニアの**六十進法**が知られている. またアメリカ大陸の固有の文明ではアステカとマヤは**二十進法**, インカは十進法である. インカには文字がなかったが, 数は紐の結び目で記録していた. 紐の結び目で数を表すのを結縄(けつじょう)という. 結縄はインカだけでなく古代中国でも使われていたほか, 沖縄や太平洋の島々にも見られる.

タワンティンスーユ (いわゆるインカ帝国) では結縄をキープ (quipu, khipu) といい, これを専門に扱う役人がいて, 人口調査や食糧備蓄倉庫の在庫管理などをおこなっていたことが知られている[2].

いまは十進法が世界の主流になっているけれども, 六十進法は時間の単位などに残っている. ヨーロッパにも十二や二十を単位とする数え方の名残と思われるものが見られる. 例えば 12 個をまとめて 1 ダースとする数え方がある. また英語の数詞は 13 から 19 までは十進法の考えに従って接尾語 "-teen" で作るのに対して, 11 と 12 は別の形になっている.

問 1. 二十を単位とする考え方について, 英語の score やフランス語の vingt などについて調べてみよ. □

問 2. いろいろな言語の命数法 (数詞の仕組み) について調べよ. □

問 3. 十進法以外の数え方の現代社会に残っている例をあげよ. □

[1] Base. 基数とも呼ぶけれども, 基数には cardinal number の意味もあり紛らわしい.

[2] F. ピース, 増田 義郎(よしお): "**図説 インカ帝国**" (小学館, 1988), pp. 163–165 ではキープは数の記録にも計算にも使われていたとしているが, ジョージ G ジョーゼフ (垣田 高夫, 大町 比佐栄 訳): "**非ヨーロッパ起源の数学** — もう一つの数学史 —" (ブルーバックス B-1120, 講談社, 1996) p. 65 ではキープは記録だけに使われ計算は別の手段によっていたと述べている.

問 4. 古代エジプトの数字について調べよ. □

問 5. バビロニアの数字について調べよ. □

古代エジプトの数字やローマ数字は十進法であるが, アラビア数字[3]とは仕組みが違う. 例えばローマ数字はつぎの表の下の段のとおり 10 の累乗とその 5 倍とをそれぞれアルファベット一文字で表す:

1	5	10	50	100	500	1000
I	V	X	L	C	D	M.

そしてこれらをもちいて例えば 1 から 10 までをつぎのように表す:

I, II, III, IV, V, VI, VII, VIII, IX, X.

問 6. ローマ数字の記数法の仕組みについて調べて説明せよ. □

問 7. アラビア数字の歴史について調べよ. □

アラビア数字もローマ数字も十進法であるけれども, ローマ数字では一, 十, 百, 千など位ごとにちがう数字で表すのに対して, アラビア数字ではどの位についても同じ数字を使う. アラビア数字のような記数法を**位取り記数法**(くらいとり)という. マヤ文明では後述するとおり二十進法の位取り記数法をもちいている.

問 8. 位取り記数法には零を表す数字の必要である理由を考えよ. □

問 9. アラビア数字表記からローマ数字表記へ, あるいはその逆に, 変換するプログラムを作れ. たとえば 16 と入力すると XVI と出力し 347 と入力すると CCCXLVII と出力する, またはその逆の変換をおこなうプログラムである. なお, プログラム言語で記述して実際にコンピューターで動かしてみるのが望ましいけれども, アルゴリズムが明確に記述してさえあればプログラム言語による記述でなくてもよい. □

理論的には 10, 12, 20, 60 に限らず, 1 より大きい任意の整数の定数 p に対して p **進法**を考えることができる. すなわち, 任意の正の整数 a は

$$a = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_2 p^2 + a_1 p + a_0$$

(a_n , ... , a_0 はいずれも 0 以上 p 未満の整数で $a_n > 0$) と表せる. ただし, n は a に依存して決まる整数である. そして a_n , ... , a_0 は a に対応してただ一組に決まる. この事実に基づいて, 数の組

$$a_n, \ldots, a_0$$

によって数 a を表すことができる.

問 10. [除法の原理] b を一定の正の整数とする. 任意の整数 a に依存して, $a = bq + r$ と $0 \leq r < b$ とをみたす整数 q , r が定まる. このことを証明せよ. □

[3] 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. インドで生まれたものだがヨーロッパへはアラビアを通じて伝えられたのでアラビア数字とよばれる. 算用数字ともいう.

ヒント．a についての数学的帰納法.

問 11. [p進法の原理] p を 1 より大きい整数の定数とする．任意の正の整数 a は

$$a = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_2 p^2 + a_1 p + a_0,$$

と一意的[4]に表せる．ただし n, a_n, a_{n-1}, ..., a_2, a_1, a_0 はいずれも a に依存して定まる 0 以上の整数で,

$$0 < a_n < p; \quad a_{n-1} < p, \ \ldots, \ a_2 < p, \ a_1 < p, \ a_0 < p$$

除法の原理をもちいてこのことを証明せよ. □

ヒント．a を p で割って余りと商とを考える.

$p = 2$ の場合が**二進法** (binary) である．つまり任意の正の整数 a は

$$a = 2^n + 2^{n-1} a_{n-1} + \ldots + 4a_2 + 2a_1 + a_0,$$

と一意的に表せる．ただし n は 0 以上の整数で, a_{n-1}, ..., a_2, a_1, a_0 はいずれも 0 または 1 である.

電子計算機の内部ではおもに二進法が使われる．二進法の位取り記数法では数字が 0 と 1 の二つだけであり, 電気的な処理が容易になることがおもな理由の一つである．二進法を最初に理論的に考察したのはライプニッツ[5]である．同じころ日本では, 二進法の考えをもちいた数学パズルなどが楽しまれていたが, 理論的な研究には至らなかった．また, 厳密な意味での二進法とはちがうけれども, 2 を単位としてまとめる考えがいくつかの言語の数詞に見られる．オーストラリア原住民の言語には 1 と 2 を意味する数詞だけを基本としている命数法がある[6]．また, 日本語の数詞についてつぎのとおり二倍になると子音はそのままで母音だけの交替するのが知られている:

1	ひと	pitö	2	ふた	puta
3	み	mi	6	む	mu
4	よ	yö	8	や	ya

注．ローマ字書きは古語．ö はオ段乙類母音を表す．昔の日本語はハ行の子音は [p] であり, 母音 a, i, u, e, o (オ段甲類), ö (オ段乙類) の六つあるいは a, i, ï, u, e, ë, o, ö の八つであった．なお古代日本語に a とö との母音交替の例はこの表のもののほかにも多い.

十進法から二進法に換算するにはつぎのようにする．十進表示の整数が与えられたとして, 2 で割ることをくり返す．商と余りとを求める整数の割算を, 商が 0 になるまで反復し, 得られた余りの列を逆順に読むと, 与えられた数の二進表示になる.

---

[4] 一通りに決まること.

[5] G. W. von Leibniz, 1646–1716.

[6] ジョージ・G・ジョーゼフ“非ヨーロッパ起源の数学”, p. 70 参照.

例 1. 十進法の 377 を二進法に換算する:

377	
188	...1 ↑
94	...0 |
47	...0 |
23	...1 |
11	...1 |
5	...1 |
2	...1 |
1	...0 |
0	...1 |

ゆえに十進法の 377 は二進法で 101111001 である. □

問 12. この方法で正しい結果の得られる理由を考えよ. □

問 13. 十進法の 100 を二進法で表せ. □

問 14. 大きい整数を二進法で表すと, 同じ数を十進法で表したときとくらべて, 桁数はほぼ 3.3 倍になる. なぜか. □

逆に二進法から十進法に換算するには 2 の累乗の表

$$2^0 = 1,\ 2^1 = 2,\ 2^2 = 4,\ 2^3 = 8,\ 2^4 = 16, \ldots$$

を用意しておいて, 例えば二進法の 110101 は

$$1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$$
$$2^5 + 2^4 + 2^2 + 2^0 = 32 + 16 + 4 + 1 = 53$$

のようにすればよい.

問 15. 二進法から十進法への, 別の換算法を考えてみよ. □

p 進法は 1, p, p^2, p^3, ... を単位にして数える仕組みであるが, そのほかに 1, 5, 10, 50, 100, ... を単位に数える**二・五進法**(にごしんほう) (biquinary, 二進五進法) もある. 例えば, そろばんは二・五進法に基づいている.

また, マヤについて注目すべきことは, インドとは独立にゼロを発見し位取り記数法をもちいていたことである. マヤでは 1, 20, 360, 7200, 144000, ... を単位とする変則的な二十進法の位取り記数法がもちいられていた. 20 から 360 へ 18 倍するのを除いてすべて 20 倍である.

なお, 暦にはこの変則二十進法が使われたが, 商取引には純粋の二十進法を使ったらしい. このことについては 青木 晴夫: "**マヤ文明の謎**" (講談社現代新書 757, 1984), p. 112 参照. また, マヤ文明に分数がなかったという誤解については同書 p. 198 参照.

問 16. マヤ数字について調べよ. □

ここまでおもに整数について考えてきたが, 小数についても同様に十進法, 二進法などが考えられる. p を 1 より大きい整数の定数とする. $0 \leq a < 1$ をみたす任意の実数 a は

$$a = a_1 p^{-1} + a_2 p^{-2} + \ldots + a_n p^{-n} + \ldots$$

(a_1 , a_2 , ..., a_n , ... はいずれも 0 以上 p 未満の整数) と表せる. ただし, 右辺が有限項で終わるとは限らない. これが p 進法小数表示の原理である.

とくに $p = 2$ の場合を考えると, $0 \leq a < 1$ をみたす任意の実数 a は

$$a = \frac{1}{2}a_1 + \frac{1}{4}a_2 + \ldots + \frac{1}{2^n}a_n + \ldots$$

(a_1 , a_2 , ..., a_n , ... はいずれも 0 または 1) と表せる. これが二進小数である.

例 2. $3/4$ を二進法で表すと 0.11 である. □

例 3. $1/3$ を二進法で表すと循環無限小数 $0.010101\ldots$ である. □

問 17. $2/5$ を二進法の小数で表せ. □

負でない任意の実数 a は $a = b + c$ (b は負でない整数, $0 \leq c < 1$) と整数部分と小数部分とに分けることができるから, b と c とをそれぞれ p 進法で表して小数点の左と右に記せば a が p 進法で表せる.

例 4. 円周率 π を二進法で表すと $11.00100100001111111011010101\ldots$ (循環しない無限小数) である. □

負の数を表すには, 絶対値と負の符号による日常的な表し方が, 十進法以外にもそのまま適用できる.

例 5. 十進法の -5 , -1.25 をそれぞれ二進法で表すと -101 , -1.01 である. □

コンピューターの中で負の数を二進法で表すときは, 絶対値と負の符号による表し方とは別の“補数表示”がよくもちいられる. 補数表示にもちいられる**補数** (complement) には詳しくは“2 の補数”と“1 の補数”があり, 単に補数と言えば通常は 2 の補数を意味する.

問 18. 補数表示について調べよ. □

2　ディジタルとアナログ

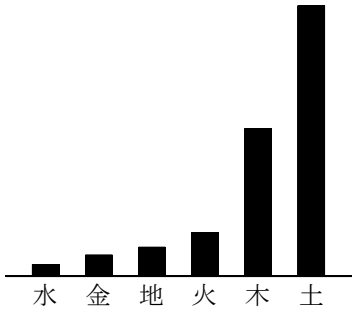
近ごろディジタル, アナログと言う語を至る所で耳にするが, 言葉の広まっている割にはこれらの概念は正しく理解されていない. 多く見られる誤解は, ディジタルはハイテクであるというものである. またディジタルを二進法と思いこんだりアナログ・ディジタルの概念を連続・離散の概念と混同したりする誤りも多い. ディジタル, アナログの概念の正しい理解がたいせつである.

数値を表すのに数字などを連ねて表す**ディジタル**な表し方のほかに, 数値に対応する (例えば比例する) 量によって表す**アナログ**な表し方がある. 数を表示したり記録したり伝えたり計算したりするのに, アナログとディジタルとのどちらの表し方も使われる.

例えば時計のアナログとディジタルとを比較してみる. これは表示の違いで, アナログ時計では針の位置で時刻を示すのに対して, ディジタル時計では数字を連ねて時刻を示す.

つぎの表とグラフとは同じデータ[7]をディジタルとアナログで表したものである. なお, グラフの目盛は印刷の都合で省略してある. 表はディジタルでありグラフはアナログである.

	($\times 10^6$km)
水	57.9
金	108.2
地	149.6
火	227.9
木	778.3
土	1429.4



表では数値が数字で表してあるのに対して, グラフでは棒の高さを数値に比例させて表してある.

問 19. 上の例のようなディジタルとアナログの表し方について, それぞれにどんな特徴があるか考えよ. □

計算器具のディジタルのものにはそろばんなどがあり, アナログのものとしては計算尺[8]がある. そろばんの例でわかるように, ディジタル表示に用いられるのは狭い意味の数字に限らず, 記号などさまざまのものがある. 計算機など電子機器では電圧や電流の有無, 電圧の $+$ と $-$, 磁化の向き (N か S か) などがもちいられる. ディジタル (digital) の語源は指を意味するラテン語 digitus で, 指で数えることに由来する. アナログの語源は比例, 相似などを意味するギリシャ語 analogía である.

[7] 太陽から惑星までの距離.
[8] 対数目盛を刻んだ二つの物差しをたがいにすべらせて乗除算などをおこなう器具.

問 20. 計算尺とはどんなものか調べてみよ. □

アナログとディジタルとを数値の表し方の違いとして説明したけれども, 数以外の情報についても表し方にアナログとディジタルが考えられる. 例えば音は空気の波であり, 時刻につれて変化する空気の圧力の数値をアナログで表すかディジタルで表すかという表し方の違いがある.

音楽レコードにアナログ盤 (LP など) とディジタル盤 (CD など) とがある. これは情報の記録の仕方の違いで, LP では音波を電気信号に変えてほぼそのままの波形[9]を溝に刻んであるのに対して, CD では音波を電気信号に変えたあと電圧を毎秒 44,100 回測定しその値を二進法 16 桁で表してその数字の列を記録してある. つまり CD に録音するときにはアナログな情報をディジタルに変える AD変換をおこない, 再生するときにはディジタル情報をアナログに変える DA変換をおこなっている.

電気信号に変換した音声を伝送するのに, 音波の形に対応する電気の波をそのまま送るアナログ伝送と, CD の記録方式と同様に一定時間ごとに測定した電圧を数字の列で表して (つまり AD変換して) その数字の列を電気的に送るディジタル伝送とがある. 前者の例には固定電話, ラジオ放送, 地上波のテレビ放送などがあり, 後者の例には携帯電話, テレビの衛星放送の音声などがある. 映像にもアナログ伝送とディジタル伝送とがあり, いわゆるディジタルテレビでは映像も音声もディジタル伝送であるが, それ以外のテレビ放送では地上波でも衛星でも映像はアナログ伝送である.

さて, **離散** (discrete) と**連続** (continuous) という概念がある. 個数や場合の数のように一つずつ数えることのできる量は離散である. つまり, 整数で量れる量は離散である. 一方, 長さや重さや時間などのように, いくらでも細かいちがいのある量は連続である. 連続であることの厳密な定義は難しいので省くが, 実数で表される量が連続である.

アナログな表し方では原理的に連続な量が表せる. これに対して, ディジタルで連続な量を表そうとすると無限の桁数を要し, 理論上は可能であるけれども, 実用上は連続な量の近似しか表せない. なぜなら有限の一定の桁数でディジタルに表せるのは離散な量であり, 連続な量は離散な量によって近似的に表すことになるからである. この事実が誤解されて, アナログ・ディジタルの概念と連続・離散の概念との混同を招いている.

実用上は, アナログで表すには精度の技術的な限界の問題があり, ディジタルで桁数を多くとる方がアナログよりも連続な量を精度よく近似できる. またアナログに記録してもとと同じ情報を完全に復元することは不可能であるのに反し, ディジタルに記録した情報は事実上もとどおりに復元[10] することが可能である. 伝送についても同様である. 例えば CD の再生装置には, ディスク上のディジタル情報を読み誤ったときに自動的に訂正する仕組みがあり,

[9] 厳密に言うと, もとの波形のままでなく, ある変換をアナログ的に施している.
[10] 正確にいえば, もとの情報との違いのある確率が極めて小さい.

録音されたときと同一のディジタル情報を復元したあとでこれを DA変換している.

ディジタル情報を伝送するときあるいは記録されたディジタル情報を再生するときの誤りを防ぐためには, 情報に検査用の桁をつけ加えた“誤り訂正符号” (error-correcting code, ECC) と言う符号化 (コード化) をもちいる. ディジタル情報の一部の桁に読み誤りや欠落があっても, 誤り訂正符号をもとにして計算しもとの情報を理論的に求めることができる.

誤りを検出するだけならばパリティ (parity) をもちいてできる. これは ECC による誤りの自動訂正よりも簡単な仕組みで実現できる. 誤りが偶発的であれば, 誤りの検出された所をふたたび読み直してみることで正しい情報を得られる. この方式も多く実用されている.

問 21. パリティとそれをもちいた誤り検出について調べよ. □

問 22. 誤り訂正理論について調べよ. □

ディジタルについて多く見られるもう一つの誤解は, ディジタルでは 0 と 1 だけで情報を表すという思い違いである. すでに例を挙げてきたとおり, ディジタルが二進法であるとは限らないのはもちろんである. 電子工学技術によるディジタル情報処理に主に二進法が使われることから生まれる誤解であろう.

記録や伝送にディジタルのすぐれていることを述べてきたが, 計算などの情報処理にもアナログよりディジタルのすぐれた点が多い. ただしディジタルな処理には大量の情報を高速に扱う必要がある. 例えば CD 1 枚には数十億ビット[11] の情報が記録してあり, 再生するときは毎秒百数十万ビットを読み出さねばならない.

高速に大量のディジタル情報の処理できる電子工学技術ができて以来, 電子計算機は専らディジタルのものになっている. アナログ式の電子計算機もかつて作られたことがあるけれども, 精度の限界の問題や複雑な計算の困難であることなどから, 特殊な目的以外にはほとんど使われなくなった. 従って, 電子計算機といえばいまはディジタル電子計算機を意味する.

問 23. 具体例を挙げてアナログとディジタルについて説明しその違いについて考えよ. □

問 24. 連続と離散について具体的に説明せよ. □

問 25. ディジタルと離散, ディジタルと二進法などの混同される理由について, よく考えてみよ. □

[11] 二進法の桁をビット (bit) という. 後述.

3　二進法による計算

二進法の加減乗除の計算は, 筆算でも珠算[12]でも, 二進法一桁の加算, 乗算をもとに, 十進法の場合と同様に考えておこなえばよい. 加算の“九々”は

$$0+0=0,\quad 0+1=1,\quad 1+0=1,\quad 1+1=10$$

また乗算の“九々”は

$$0\times 0=0,\quad 0\times 1=0,\quad 1\times 0=0,\quad 1\times 1=1$$

である. 一桁の足し算と**くり上がり**[13] (carry) の処理をくり返せば何桁の足し算でもできる. 足し算と, 一桁の掛け算と, 位取りを変えることをくり返せば何桁の掛け算でもできる. 引き算, 割り算についても類推できるであろう.

問 26. 加減乗除の計算を二進法で実行してみよ. □

問 27. 二進法の加減乗除の計算の仕方について説明せよ. □

二進法の減算は, 十進法の筆算や珠算の減算のやり方にならっても出来るが, 計算機の内部ではつぎのような方法がもちいられる. 桁数は一定に固定しておく. 二進法で表した x, y が与えられたとき $x-y$ を計算するには, まず y の各桁の数字を 0 は 1 に変え 1 は 0 に変え[14], その結果を $\overline{y}$ とすると, $x+\overline{y}+1$ を, 最上位からのくり上がりは無視して計算する. なお, $\overline{y}+1$ を y の**補数** (complement) という. 例えば $12-7$ を 4 桁で考えてみると, 12 は二進法で 1100 , 7 は二進法で 0111 でありこれを反転すると 1000 になる. 最上位 (8 の位) からのくり上がりを無視して

$$1100+1000+0001$$

を計算すると 0101 , つまり十進法の 5 である.

問 28. この計算法で差が正しく得られる理由を説明せよ. □

二進法では, 十進法で計算するのとくらべて一桁ずつの計算が著しく簡単になる. そのかわり桁数が十進表示の場合より多くなるから, くり返しの回数は多くなる. 機械で処理するには, 複雑なことを一回おこなうよりも, 単純なことを多くくり返す方がふさわしい.

十進法で大きな数を記述するときに三桁ごとにコンマで区切ることが多い. 二進法で表した数も同じように三桁ごとに区切り, 三桁のまとまりをひとつの数字で表せば, 八進法 (octal) の表示になる[15].

問 29. そのことについて説明せよ. □

例 6. 二進法の 1011101 を $1,011,101$ と三桁ごとに区切って八進表示 135 が得られる. □

[12] そろばんによる計算. そろばんをもちいて二進法で計算するには五の珠だけを (一の珠とみなして) もちいるとよい.

[13] 計算機業界では**桁上げ**という.

[14] 反転という.

[15] 八進法の数字としてはアラビア数字のうち $0-7$ を使えばよい.

二進法で表した数を四桁ごとに区切れば十六進法 (hexadecimal) になる. 十六進法では十五までの数をそれぞれ一個の数字で表す必要がある. 十進法の 0 – 9 に当たる十六進数字はアラビア数字をそのまま用い十進法の 10 – 15 に当たる十六進数字としては A–F を当てるのが現在の習慣である. すなわち, 十六進数字として

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ \mathrm{A},\ \mathrm{B},\ \mathrm{C},\ \mathrm{D},\ \mathrm{E},\ \mathrm{F}$$

をもちいる.

例 7. 二進法の 1010 1100 1110 0001 は十六進法で ACE1 である. □

十進法の一桁ずつを二進法四桁で表すのを二進化十進法 (binary coded decimal, BCD) という. 例えば十進法の 37 は二進法では 100101 であるが, 二進化十進法では 00110111 である. 電卓 (電子式卓上計算器) の内部では二進化十進法がもちいられる. コンピューターの内部ではおもに二進法がもちいられ一部の処理について二進化十進法がもちいられる.

問 30. 計算や記録に関して二進法と二進化十進法との優劣を比較せよ. □

問 31. コンピューターでもちいられる BCD は, さらに細かく分類すると, ゾーン十進法 (zoned decimal) とパック十進法 (packed decimal) との二種類がある. これらについて調べよ. □

二進法の一桁を**ビット** (bit) という. ビットは 0 と 1 とのいずれかであり, 二つの状態例えば電流の有無の区別として表すことができる. ビットは二進法の一桁であるとともにまた離散的な情報の最小単位でもある. 従って情報量を量る単位としてもビットがもちいられる.

コンピューターの中では数値情報以外のもろもろの情報もビット列として表して処理する. 文字などに対してはビット列としての表し方 (コード) を定めておく. 例えば, つぎの表はラテン文字とアラビア数字などを 8 ビットで表すコード表の一部である.

文字	コード
...	...
0	0011 0000
1	0011 0001
...	...
9	0011 1001
...	...

文字	コード
...	...
A	0100 0001
B	0100 0010
...	...
Z	0101 1010
...	...

文字	コード
...	...
a	0110 0001
b	0110 0010
...	...
z	0111 1010
...	...

問 32. 文字コードについて調べよ. □

なお, コンピューターでは 8 ビットずつまとめて処理するのが一般的であり, 8 ビットを**バイト** (byte) とよぶ. 情報量の単位としてのビット, バイトをそれぞれ b, B と記す[16]. なお, メガビット, キロバイトなどビットやバイトと

[16]遺伝情報に関しては b がビットでなく塩基を意味するので注意. DNA では A, C, G, T の文字で記される 4 種類の塩基 (base) の一個が 1b であり, ほぼ 2 ビットの情報量に相当する. なお, 塩基対を意味する bp という単位も使われる.

いうコンピューター関係の単位に接頭語キロやメガなどを使うときに, キロ (kilo, k) は 10^3 倍と 2^{10} 倍との二とおり, メガ (mega, M) は 10^6 倍と 2^{20} 倍との二とおり, ギガ (giga, G) は 10^9 倍と 2^{30} 倍との二とおりに使われるので, 注意を要する. 例えば 1キロバイトは 1000バイトの意味と 1024バイトの意味と, 二とおりに使われる. また, 一部には 10^3 倍を小文字 k で, 2^{10} 倍を大文字 K で表す習慣もある. ただしメガは 10^6 倍でも 2^{20} 倍でも大文字 M を書く. 本来のキロ, メガ, ギガなどはメートル法など SI 単位系でそれぞれ 10^3 倍, 10^6 倍, 10^9 倍などを意味する接頭語である.

問 33. ビット, バイトなどの用語の歴史について調べよ. □

問 34. SI 単位系の接頭語キロ, メガなどについて調べよ. □

問 35. いろいろな情報のコード化について調べよ. □

問 36. 遺伝子コードについて調べよ. □

4　命題論理, 組み合わせ論理回路

ビットは命題の**真理値** (truth value) に対応させることができる. この考え方で, 二進法の数値演算は論理演算 (命題演算) に還元することができる. 基本的な命題演算をおこなう電子回路を作れば, それを組み合わせて二進法の計算などをおこなう回路が得られる.

命題の真理値 **真** (true), **偽** (false) をそれぞれ 1 , 0 で表す[17]. 命題の基本的な論理演算として ¬ (not), ∧ (and), ∨ (or) の三つを考える. ¬ は単項演算で $\neg A$ は A の否定 "A でない" を表す. ∧ と ∨ とは二項演算で $A \wedge B$ は "A かつ B", $A \vee B$ は "A または B" を表す. ただし ∨ は inclusive, つまり A も B もともに真であるときに $A \vee B$ は真と考えることに注意. おのおのの命題演算の意味は正確にはつぎの真理値表で定められる.

A	$\neg A$
1	0
0	1

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

二進法の計算をおこなう電子回路を考えるときにもちいる論理は, 数理論理学の中で**古典命題論理** (classical propositional logic) とよばれるものである. **古典論理**では命題は真・偽いずれかの真理値をとると考える[18]. **命題論理**ではいくつかの命題から命題を作る論理演算だけを考える. なお, 数理論理学でおもに扱われるのは命題論理よりも広い**述語論理**(predicate logic) である. 述語論理では述語 (命題函数) を考え, 命題論理の論理演算だけでなく, 例えば"あらゆる自然数 n について ..."のような全称命題を作る論理演算や, "...をみたす実数が存在する"のような存在命題を作る論理演算も扱う.

論理記号は, 数学の他の分野の記号とくらべて, 統一されていない. つぎにあげるように, 同じ意味にさまざまな論理記号が使われているので, 文献を参照するときには注意を要する:

not A	$\neg A$, $\rightharpoondown A$, $\sim A$, $\overline{A}$, A'
A and B	$A \wedge B$, $A\&B$, $A \cdot B$, $A.B$
A or B	$A \vee B$, AB, $A + B$
A implies B	$A \supset B$, $A \to B$, $A \Rightarrow B$

古典命題論理の命題演算はすべて not (¬), and (∧), or (∨) の三種類の命題演算を組み合わせて表すことができる[19].

[17] 現代の数理論理学ではそれぞれ $\top$, $\bot$ で表すことが多い.

[18] その意味で古典論理を二値論理 (two-valued logic) ともいう. これに対して非古典論理 (nonclassical logics) と総称される諸々の論理がある.

[19] 実は ¬ と ∧ だけでもよい. また ¬ と ∨ だけでもよい.

例 8. $(\neg A)\vee B$ と $\neg(A\wedge(\neg B))$ とは同値である. それを示すには次のように真理値表を書いて計算すればいい.

A	B	$\neg A$	$(\neg A)\vee B$	$\neg B$	$A\wedge(\neg B)$	$\neg(A\wedge(\neg B))$
1	1	0	1	0	0	1
1	0	0	0	1	1	0
0	1	1	1	0	0	1
0	0	1	1	1	0	1

□

$(\neg A)\vee B$ を $A\supset B$ と書く.

問 37. $\neg(\neg A)$ が A と同値であることを示せ. □

問 38. $A\vee(A\wedge B)$ が A と同値であることを示せ. □

問 39. $A\supset(A\supset B)$ が $A\supset B$ と同値であることを示せ. □

問 40. $A\supset(A\vee B)$ の真理値がつねに 1 であることを示せ. □

上の問題の論理式のように, 真理値がつねに 1 になることを, 論理式が**恒真** (valid) であるという.

問 41. 恒真な論理式の例をいくつかみつけよ. □

問 42. $\neg(\neg A\wedge\neg B)$ が $A\vee B$ と同値であることを示せ. □

注. 問題の左の式は $\neg\big((\neg A)\wedge(\neg B)\big)$ の意味である. 括弧(かっこ)を節約するため, $\neg$ は $\wedge$, $\vee$ よりも優先するという規約を設ける (数式について乗算 $\times$ が加減算 $+$, $-$ よりも優先するという規則を設けて括弧を節約するのと同様である). また $\wedge$ について結合律がなりたつ, つまり $(A\wedge B)\wedge C$ と $A\wedge(B\wedge C)$ とはつねに同値である. このことを根拠に括弧を省いて $A\wedge B\wedge C$ と書く. $\vee$ についても同様. なお, $\wedge$ と $\vee$ との優先順位は区別しない. 従って $A\wedge B\vee C$ のような書き方はあいまいであるから避ける.

問 43. $\wedge$, $\vee$ について結合律のなりたつことを示せ. □

問 44. $(A\wedge\neg B)\vee(\neg A\wedge B)$ と $(A\vee B)\wedge\neg(A\wedge B)$ とが同値であることを示せ. □

$(A\wedge\neg B)\vee(\neg A\wedge B)$ で表される命題演算を A と B との exclusive or (排他的論理和) とよび, $A\veebar B$ または $A\oplus B$ などと書く. $\veebar$ に関しても結合律がなりたつから $A\veebar B\veebar C$ などの書き方が許される.

問 45. $\veebar$ について結合律のなりたつことを示せ. □

問 46. 命題論理について調べよ. □

ここでは 0 と 1 との演算を古典命題論理の論理演算で説明してきたが, ブール[20]の創めたブール代数 (Boolean algebra) という代数系の中でとくに**二値ブール代数**または**二元ブール代数**とよばれるものをもちいても実質的に同じことである. スイッチを組み合わせた二進演算回路などを考察するのに二値ブール代数をもちいることはシャノン[21]の創案である.

ブール代数というのは“論理の代数”として考案された, つぎのような代数系である. 集合 L に少なくとも 0, 1 の二つの要素があり, join と呼ぶ演算

[20] George Boole, 1815–1864.
[21] Claude E. Shannon, 1916–2001.

$a \cup b$ と meet と呼ぶ演算 $a \cap b$ と, complement[22] と呼ぶ演算 a' が L にあってつぎの公理がすべてみたされるとき, 集合 L はブール代数であるという. とくに, L の要素が 0 と 1 の二つだけであるとき, L を二値ブール代数という.

$$
\begin{array}{cc}
a \cup a = a & a \cap a = a \quad \text{(冪等律)} \\
a \cup b = b \cup a & a \cap b = b \cap a \quad \text{(交換律)} \\
(a \cup b) \cup c = a \cup (b \cup c) & (a \cap b) \cap c = a \cap (b \cap c) \quad \text{(結合律)} \\
a \cup (b \cap c) = (a \cup b) \cap (a \cup c) & a \cap (b \cup c) = (a \cap b) \cup (a \cap c) \quad \text{(分配律)} \\
(a \cup b) \cap b = b & (a \cap b) \cup b = b \quad \text{(吸収律)} \\
a \cup a' = 1 & a \cap a' = 0 \quad \text{(補元律)}
\end{array}
$$

現代数学的に言えばブール代数は束 (lattice) という代数構造の一種で可補分配束 (complemented distributive lattice) とよばれるものに当たる.

ブール代数では a と b の meet $a \cap b$ を $a \cdot b$ または ab とも書いて, a と b の積 (product) とも呼ぶ. また $(a \cap b') \cup (a' \cap b)$ を $a + b$ と書き[23], a と b の和 (sum) と呼ぶことがある.

ブール代数では, つぎのことが成り立つ.

$$a \cup 0 = a,\ a \cap 0 = 0,\ a \cup 1 = 1,\ a \cap 1 = a,\ a'' = a,\ 0' = 1,\ 1' = 0.$$

命題論理の真, 偽, and, or, not をそれぞれ二値ブール代数の 1, 0, meet, join, complement に対応させると, 論理を代数的に扱うことができる.

問 47. ブール代数について調べよ. □

問 48. シャノンの経歴や業績について調べよ. □

電気的には例えば電圧の加わっている状態あるいは電流の流れている状態で 1 を, 電圧あるいは電流のない状態で 0 を表す. 基本的な命題演算 $\neg$, $\wedge$, $\vee$ はトランジスタなどの素子をもちいて実現できる. トランジスタは半導体の結晶の中の電子の動きの原理に基いた素子であって, 電流を電流で制御する (または電流を電圧で制御する) 働きがあり, 電子的なスイッチなどとして使われる.

入力端子に電圧を加えないときに出力端子に電圧の現れる NOT 回路, 二つの入力端子の両方に電圧を加えたときに出力端子に電圧の現れる AND 回路, 二つの入力端子の少なくとも一方に電圧を加えたときに出力端子に電圧の現れる OR 回路の三種類に論理回路がトランジスタをもちいて容易に作れる. この三種類の論理回路を要素として, 必要な論理演算をおこなう論理回路を構成する[24].

---

[22] 補元. 二進法補数表示に使われる補数と同じ英語だが, 違う概念である.

[23] なお join $a \cup b$ を $a + b$ と書き complement a' を $-a$ と書く流儀もある.

[24] 実際には電子工学上の技術的な理由から $\neg(X \wedge Y)$ の演算をおこなう NAND 回路と $\neg(X \vee Y)$ の演算をおこなう NOR 回路とを基本にすることもある.

5　論理回路による計算

さて, 二進法一桁の足し算を命題演算で表すことを考えてみる. 一桁の二つの数の和は $1+1=10$ のように二桁になる場合があるので, 和をその桁の数字 S と上の桁へのくり上がり C とにわけて考える.

X	Y	C	S
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

左の表から $C = X \wedge Y$, $S = X \veebar Y = (X \wedge \neg Y) \vee (\neg X \wedge Y)$. これを実現するには NOT 回路を二つ, AND 回路を三つ, OR 回路を一つもちいる. 二つの入力端子 X, Y への入力に対応して左の表のとおりの出力 S, C を得る回路を**半加算器** (half adder) という.

問 49. 半加算器の回路を考えよ. □

半加算器は一桁の加算をおこない, 上の桁へのくり上がりを出すことはできるが, 下の桁からのくり上がりを受け取ることができない[25]. その桁の二つの数字 X, Y のほかに下の桁からくり上がって来る Z も考えると, 一桁の加算にはつぎの表の論理演算をおこなうことが必要である. この表の演算をおこなう回路を**全加算器** (full adder) という.

X	Y	Z	C	S
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

$S = X \veebar Y \veebar Z$ は容易にわかるであろう. くり上がり C は X, Y, Z の三者多数決 $(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ である. 全加算器を NOT, AND, OR の基本回路から直接に組むよりも, 半加算器を二つと OR 回路を一つもちいて作るのが簡単である.

半加算器をおのおの HA_1 と HA_2 とする. X と Y とを HA_1 の入力として, HA_1 の S 出力と Z とを HA_2 の入力とする. HA_2 の S 出力を全加算器の S 出力とする. また HA_1 の C 出力と HA_2 の C 出力とを OR 回路の入力につなぎ, その OR 回路の出力を全加算器の出力とする.

上に説明した全加算器の回路では $(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ と $(X \wedge Y) \vee ((X \veebar Y) \wedge Z)$ との同値であることをもちいている.

問 50. その同値であることを示せ. □

たとえば 8 ビットの加算をおこなうには全加算器を 8 個用意して, おのおのの C 出力を一つ上の桁の Z 入力に接続すればよい (加算だけを考えれば最下位だけはくり上がりが入ってこないから半加算器でもよいが, 実際のコンピューターでは減算などにも同じ回路を兼用するため最下位も全加算器とする). この 8 ビット加算回路に否定回路などを追加して 8 ビットの減算回路が作れる. 加算回路と, 数を保持する回路 (置数器, register) と, 桁をずらす (shift する) 回路とを組み合わせて乗算回路ができる.

ここまでに述べてきた仕組みを組み合わせて, 電卓を作ることができる. しかし, コンピューターを作るにはこれだけでは足りない.

[25] **半**加算器の名はそのことによる.

6　逐次制御, 自動計算, プログラム

自動制御の考えの一つに逐次制御 (sequence control) がある. いくつかの操作をあらかじめ定められた順序に従ってつぎつぎとおこなう仕組みである. 逐次制御の機械の方式として, 操作の手順を機械の部品で決める方式と, 操作の手順を記述した「プログラム」を機械が解読して動作する方式とがある. 後者を逐次制御の中でもとくにプログラム制御という.

問 51. 電気器具などで逐次制御のおこなわれている例を挙げよ. □

問 52. 逐次制御について調べよ. □

プログラム制御の自動計算機を作ろうと初めて試みたのはバベッジ[26]である. 歯車などを組み合わせた十進演算装置をプログラムで逐次制御して, 函数表[27]の作成などを自動的におこなうことをめざした“解析機関”(analytical engine) 考案し, 1833年頃から組み立て始めたが, ついに完成しなかった. まだ電気のない時代で, 動力には蒸気機関をもちいる計画であった. プログラムにはジャカール織機[28]のためのパンチカードをもちいた.

問 53. ジャカール織機とはどんなものか. □

問 54. バベッジの差分機関 (difference engine, 階差機関) と解析機関について調べよ. □

問 55. バベッジの経歴や業績について調べよ. □

それから一世紀以上を経たのち, 第二次世界大戦で弾道計算や暗号解読など軍事目的で大量の複雑な計算を高速におこなう需要が生じてきて, 電気的あるいは電子的な自動計算機が計画された. 軍事機密もあって当時のくわしい歴史はわからないが, 公開されていてよく知られていることだけを紹介する. 技術的背景としては, バベッジのころと違って電灯, 電話, 無線通信, ラジオ放送, レーダーなどがすでに実用化されていて, 継電器[29]はもとより, 電子的な素子として真空管が普及していた.

しばしば最初の電子計算機とみなされているアメリカのペンシルバニア大学の ENIAC は 1943年にエッカートとモークリーによって開発が始められ, 遅くとも 1945年には稼働したらしい. トランジスタのなかった時代なので素子には真空管 18,000 本をもちい, 消費電力 140kW, 長さ 30m × 高さ 3m × 奥行き 0.9m という巨大なものであった. “弾丸の飛ぶより早く弾道を計算する”といわれたが, 記憶容量は小さく, プログラムには配線盤をもちいていたので, プログラムの設定や点検・修正に多大の時間を要したという. なお, 計算や記憶はすべて十進法でおこなわれていた. ENIAC は電子的なプログラム制御自動計算機とはいえ, 今日のコンピューターとは原理が異なるので, これを最初の電子計算機と考えるのは適当でないと思う.

---

[26] Charles Babbage, 1792?–1871.

[27] 函数 (かんすう) は関数と同じ意味. 中国で function の音意訳として函数 (hánshù) という語が作られ, 日本でもこれを借用していたが, 太平洋戦争後の漢字制限に伴い関数という当て字が作られた.

[28] ジャカード織機ともいう. ジャカール (J. M. Jacquard, 1752–1834) の完成した紋織機.

[29] リレー (relay) ともいう. 電磁石でスイッチを操作する素子.

問 56. ENIAC について調べよ. □

さて, 1946年に数学者フォン・ノイマン[30]は, 二つの提案を発表した. 第一は二進法で計算をおこなうことである. 第二はプログラムをコード化してデータと同様に扱うことである. コード化したプログラムを記憶装置に記憶させておき, プログラムの一つ一つの命令を機械が自ら記憶装置から読み出して実行していく**プログラム内蔵** (stored program) あるいはプログラム記憶とよばれる制御方式である. この第二の提案はとくに重要で, **計算可能性**の理論に基づいている.

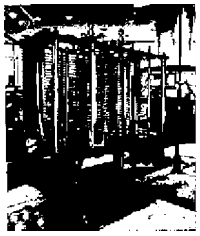
フォン・ノイマンの方式による最初の計算機は 1949年から稼働したイギリスのケンブリッジ大学の EDSAC である. その後ペンシルバニア大学でも EDVAC が稼働した. 現在のコンピューターはすべてプログラム内蔵方式つまりフォン・ノイマンの方式である.

問 57. 計算用具, 計算機などの歴史を調べよ. □

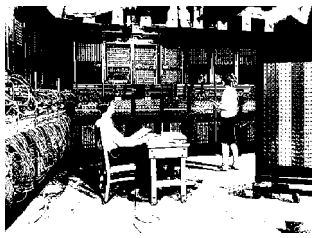
問 58. 世界のまたは日本の初期のコンピューターについて調べよ. □

問 59. フォン・ノイマンの経歴や業績について調べよ. □

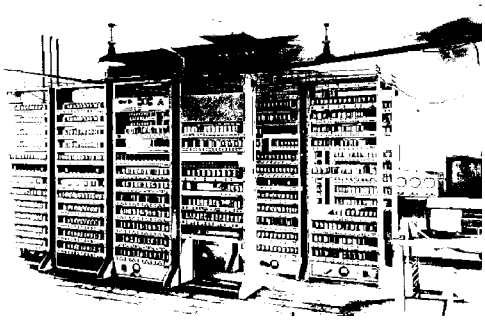
問 60. フォン・ノイマンがどうしてプログラム内蔵方式を発明し得たのか, その理由を考えよ. □



解析機関



ENIAC



EDSAC

[30] J. von Neumann, 1903–1957.

7　計算可能性の概念

数学の中に集合論という分野があるが, 集合論に逆理が発見されるという衝撃的な事件があり, 数学の第三の危機とよばれている. 集合論に矛盾の含まれているのがわかったことを契機に, 信頼できる数学の再建をめざして数学の論理的基礎を探求する数理論理学 (数学基礎論) の研究が始められた. 証明とは何か, 計算とは何か, そういうことを探る研究の中から, まず 1931年にゲーデル[31]の不完全性定理が発見された. この定理の証明に, 計算可能性にかかわる議論がもちいられている. さらに一般に計算可能とはどういうことかという問に答えるべく, クリーネ[32]の再帰的函数[33] (recursive function) の理論, テューリング[34]の万能テューリング機械 (universal Turing machine) の理論, チャーチ[35]のラムダ計算 (lambda calculus) の理論などが 1936年に発表された. これらの理論は見かけがまったく違うにもかかわらず, いずれの意味での計算可能性も論理的に同値になることが証明された.

問 61. 集合論の逆理と, その後の数学基礎論の発展について調べよ. □

アルゴリズム (算法) による計算可能性すなわち機械的な計算可能性という概念を客観的に定義しようとするさまざまな試みが 1930年代半ばからおこなわれた. テューリングは計算するという人間の行為を分析して仮想的な計算機を創案し, その仮想機械による計算を考察した. テューリングの考えたこの機械はテューリング機械とよばれる. クリーネはエルブラン[36]とゲーデルの示唆に基づき数学的帰納法で定められる函数を考察して再帰的函数 (recursive function) の理論を創った. チャーチは函数を作り出すラムダ演算子 λ と代入の概念をもとにラムダ計算の体系[37]を構築した. “計算可能な函数”を客観的に定義しようとして考え出された概念が, 上に挙げたもののほかにもいろいろある. そして, これらのさまざまな概念は, 見かけがまったくちがうにもかかわらずたがいに同値であることが証明された.

計算可能性のさまざまな考え方のうち, テューリング機械 (Turing machine) に基くものをまず説明する. 計算の対象となるのは記号の有限列で表せる情報である. テューリング機械とはつぎのような仮想計算機の総称である. 機械 M は本体とテープとヘッドとからなる. テープは無限に長くて, ます目に区切られ, おのおののます目に書き込み得る記号 $s_0, \ldots, s_m$ が M に応じて定まっている. ヘッドの機能は, テープのます目に書いてある記号を読み取ったります目に記号を書き込んだり隣のます目へ移動したりすることである. M の本体は, とり得る内部状態 $q_0, \ldots, q_k$ が定まっていて, そのいずれかの内部状態をとっている. そして M の動作を定めるつぎのような写像

[31] Kurt Gödel, 1906–1978.
[32] S. C. Kleene, 1909–1994.
[33] 帰納的函数ともいう.
[34] A. M. Turing, 1912–1954.
[35] A. Church, 1903–1995.
[36] J. Herbrand, 1908–1931.
[37] ラムダ計算については高橋正子: 計算論 ——計算可能性とラムダ計算——を参照

が定まっていて, 一段階の動作はつぎのとおりである. ヘッドの位置しているます目に書いてある記号 s_i と本体のとっている内部状態 q_j との組に依存してつぎの三つの動作をおこなう:

1. そのます目の記号を書き替える.
2. 内部状態を変える.
3. ヘッドを右に l ますだけ移動する ($l = 1$, 0, -1).

数学的に抽象化して述べれば $M = \langle \Sigma, Q, \delta_1, \delta_2, \delta_3 \rangle$, Σ (記号の集合) と Q (内部状態の集合) とは空でない有限集合で

$$\delta_1 \colon \Sigma \times Q \to \Sigma\,, \quad \delta_2 \colon \Sigma \times Q \to Q\,, \quad \delta_3 \colon \Sigma \times Q \to \{1, 0, -1\}$$

である. M のヘッドの見ている記号が s_i であり M の本体の内部状態が q_j であるとき, テープのいま注目しているます目の記号を $\delta_1(s_i, q_j)$ に書き替え, 内部状態を $\delta_2(s_i, q_j)$ に変え, ヘッドは右に $\delta_3(s_i, q_j)$ だけ移動する.

テープのます目に何も書いてない場合を考えると煩雑になるので, 記号の一つを“空白”という記号として, 何も書いてないのは空白が書いてあるものとみなす. また, 機械が停止するというのも, テープ上の記号を書き替えず内部状態を変えずヘッドを動かさない動作のこととみなす. すなわちどの記号 s_i に対しても

$$\delta_1(s_i, q_j) = s_i\,, \quad \delta_2(s_i, q_j) = q_j\,, \quad \delta_3(s_i, q_j) = 0$$

のなりたつ内部状態 q_j を停止状態という.

さて, 計算可能とはそれを計算するテューリング機械の存在することと定義する. たとえば x の値を書いたテープをセットして起動すると有限時間の後にテープに $f(x)$ の値を書いて停止するようなテューリング機械が作れるとき, 函数 f は**計算可能** (computable) であるという[38]. コンピューターの構造をおおよそ知っているなら, テューリング機械とつぎのように対比させて考えてみるとよい: テューリング機械の本体はコンピューターの CPU[39], テューリング機械のテープはコンピューターの記憶装置 (memory) とみなす.

問 62. ゲーデルの経歴や業績について調べよ. □

問 63. チャーチの経歴や業績について調べよ. □

問 64. テューリングの経歴や業績について調べよ. □

問 65. クリーネの経歴や業績について調べよ. □

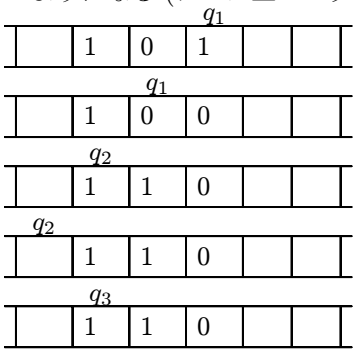
例 9. つぎに示す例は二進法で記された数に 1 を足すテューリング機械である. テープ上に二進法で数を書き, 最下位 (右端) の桁の位置にヘッドを置いて内部状態 q_1 で始動すると, 1 を足した数を書いて停止する. 記号は空白, 0, 1 の三つ, 内部状態は q_1, q_2, q_3 の三つである.

[38] 一変数で説明するが, 二変数以上でも同様.
[39] 中央処理装置, central processor unit.

	空白	0	1	説明
q_1	$(1, q_3, 0)$	$(1, q_2, 左)$	$(0, q_1, 左)$	くり上がり 1
q_2	$(空白, q_3, 右)$	$(0, q_2, 左)$	$(1, q_2, 左)$	くり上がり 0
q_3	—	—	—	停止

表には, 左の欄の内部状態と上の欄の記号との組み合わせに対する動作を示してある. たとえば q_1 の行, 0 の列にある $(1, q_2, 左)$ はテープ上の記号を 1 に書き替え内部状態を q_2 に変えたあと左へ移動することを意味する. 空欄は停止の意味.

たとえばテープに 101 (十進法の 5) を書いて動かした場合の動作はつぎのようになる (テープ上のヘッドの位置にそのときの内部状態を記してある):



□

例 10.　テープに二進法で記された数を, 空白を一つあけて右隣に書き写して停止するテューリング機械がつぎのように作れる. 記号は数を記すのに必要な空白, 0, 1 の三つと補助的に使う $\overline{0}$, $\overline{1}$ の二つとをあわせた五つとする. 内部状態は q_0, q_1, q_{20}, q_{21}, q_{30}, q_{31}, q_4 の七つとする.

	0	1	$\overline{0}$	$\overline{1}$	空白
q_0	$(q_0, 0, 左)$	$(q_0, 1, 左)$	—	—	$(q_1, 空白, 右)$
q_1	$(q_{20}, \overline{0}, 右)$	$(q_{21}, \overline{1}, 右)$	—	—	—
q_{20}	$(q_{20}, 0, 右)$	$(q_{20}, 1, 右)$	—	—	$(q_{30}, 空白, 右)$
q_{21}	$(q_{21}, 0, 右)$	$(q_{21}, 1, 右)$	—	—	$(q_{31}, 空白, 右)$
q_{30}	$(q_{30}, 0, 右)$	$(q_{30}, 1, 右)$	—	—	$(q_4, 0, 左)$
q_{31}	$(q_{31}, 0, 右)$	$(q_{31}, 1, 右)$	—	—	$(q_4, 1, 左)$
q_4	$(q_4, 0, 左)$	$(q_4, 1, 左)$	$(q_1, 0, 右)$	$(q_1, 1, 右)$	$(q_4, 空白, 左)$

空欄は停止を意味する. □

問 66. このテューリング機械の動作についてくわしく説明せよ. □

問 67. 何かの演算をおこなうテューリング機械を考えてみよ. □

8　万能テューリング機械とフォン・ノイマンの原理

テューリングはつぎに述べる万能テューリング機械の存在を数学的に証明した. 万能テューリング機械 (universal Turing machine) とは, 任意のテューリング機械の動作をまねることの出来るテューリング機械のことをいう. まねる対象となるテューリング機械の構造を一定の方法でコード化して記述したテープを与えると, 万能テューリング機械はそのテープを読みながら対象となる機械の動作を模倣する.

M_0 を万能テューリング機械の一つとすると, M_0 のコード化の規則が定まっていて, M_0 はつぎの働きをする. 任意のテューリング機械 M と, M のテープ上の記号列 x が与えられたとして, それぞれを M_0 の規則に従ってコード化して, M のコードと x のコードとを記したテープを M_0 に設定して起動すると, M が x に対しておこなう動作を M_0 が再現する.

前に例を挙げたテューリング機械はたとえば 1 を足すという計算をおこなう専用の機械であったのに対して, 万能テューリング機械には計算可能などんな函数をも計算する能力がある. なぜかというと, 函数 f が計算可能ならば定義により f を計算するテューリング機械があり, この機械の動作を万能テューリング機械にまねさせることによって f は万能テューリング機械で計算できる. この意味で万能テューリング機械は万能である.

万能テューリング機械の存在することの証明には, テューリング機械の構造をコード化する考えが重要である. このコード化の考えは, ゲーデルが不完全性定理の証明に初めてもちいたゲーデル数 (Gödel number) の概念に基く.

問 68. ゲーデルの不完全性定理とはどんな定理か. □

さて, 函数が計算可能であることは, 前述のとおり, その函数が再帰的であることと同値である. 万能テューリング機械の存在に対応する事実を再帰的函数の理論の立場から述べると, 標準型定理に相当する. ここで, 再帰的函数について説明する. なお, 再帰的函数の理論は自然数[40] 0, 1, 2, ... の上で考える. 以下の説明では函数とは自然数の函数を意味する. ただし, 部分函数 (partial function) を考える. すなわち $f(x)$ の値があらゆる自然数に対して定まっている (そのような函数は全域的 (total) であるという) 必要はなく, ある自然数に対しては f の値が定まっていて別のある自然数に対しては f の値の存在しないような函数 f も考える. 二変数以上の函数に関しても同様である. 函数が全域的であるというのは, その定義域が自然数または自然数の組の全体に一致することに他ならない.

例 11. 自然数の範囲で $x-y$ を考えると, $x \geq y$ のときには値が定まるけれど $x < y$ のときには値が存在しない. つまり 2 変数の部分函数 $x-y$ は全域的でない. □

$f(x)$ の値の定まる x の範囲を f の定義域という. 二変数以上の函数に関

[40] 歴史的には 0 の発見は正の整数よりも後とのことであるが, 理論上は 0 を自然数に含めた方が扱いやすく, 現代数学では 0 を入れる傾向にある. ここでも現代の習慣に従う.

しても同様.

例 12. S の定義域は自然数の全体である. $x+y$ の定義域は自然数の組 (x,y) の全体である. $x-y$ の定義域は $x \geq y$ をみたす自然数 x, y の組 (x,y) の全体である. □

自然数の最も基本的な演算は“次の数” (successor) である. 自然数 x の次の自然数すなわち $x+1$ を $\mathrm{S}(x)$ または x' と書く. 再帰的函数の概念をこれから段階的に定義していく. まず,

$$\mathrm{S}(x)$$

は再帰的函数である. つぎに, 定数値函数すなわち

$$f(x_1, x_2, \ldots, x_n) = c \quad (c \text{ は定数})$$

の形の函数は再帰的函数である. また, 射影函数すなわち

$$f(x_1, x_2, \ldots, x_n) = x_i \quad (i \text{ は } 1, 2, \ldots, n \text{ のいずれか})$$

の形の函数は再帰的函数である.

再帰的函数を合成した函数は再帰的函数である. すなわち, m 変数の再帰的函数 h と n 変数の再帰的函数 g_1, …, g_m から

$$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

と定めた n 変数函数 f は再帰的函数である.

1 変数の函数 f を, 与えられた 2 変数函数 g から

$$f(0) = c, \qquad f(x') = g(x, f(x))$$

と定めるのを 1 変数の**原始再帰法**[41] (primitive recursion) という. ただし c は定数. つぎに $n > 1$ のとき, n 変数の函数 f を, 与えられた $(n+1)$ 変数函数 g と $(n-1)$ 変数函数 h から

$$\begin{aligned} f(0, x_2, \ldots, x_n) &= h(x_2, \ldots, x_n), \\ f(x_1', x_2, \ldots, x_n) &= g(x_1, f(x_1, x_2, \ldots, x_n), x_2, \ldots, x_n) \end{aligned}$$

と定めるのを n 変数の**原始再帰法**という. さて, 再帰的函数から原始再帰法で定められる函数は再帰的函数である.

つぎに, n 変数函数 f を, 与えられた $(n+1)$ 変数函数 g から, $g(x_1, \ldots, x_n, y) = 0$ となる y のあるときは

$$f(x_1, \ldots, x_n) = \min\{y \,|\, g(x_1, \ldots, x_n, y) = 0\}$$

[41] 原始帰納法ともいう.

とおき, $g(x_1,\ldots,x_n,y)=0$ となる y のないときは

$$f(x_1,\ldots,x_n) \text{ の値は存在しない}$$

として定めるのを**ミュー演算**という.

なお $\min\{y|\, g(x_1,\ldots,x_n,y)=0\}$ は $g(x_1,\ldots,x_n,y)=0$ となる y のうち最小のものを意味する. 計算論では m に対応するギリシャ文字 μ (ミュー) をもちいて上のような $f(x_1,\ldots,x_n)$ を

$$\mu y[g(x_1,\ldots,x_n,y)=0]$$

と書き表す. これがミュー演算 (mu operation) という名称の由来である. さて, 再帰的函数からミュー演算で定められる函数は再帰的函数である.

例 13. $x-y=\mu z[y+z=x]$. □

例 14. $x-y=\mu z[y+z\geq x]$. □

以上のようにして得られる函数だけが再帰的函数である. すなわち, S と定値函数と射影函数から, 合成と原始再帰法とミュー演算とを何回か (0 回以上) くり返して適用して得られる函数が再帰的函数である. なお, 厳密にはここに述べたのは“ミュー再帰的函数”の定義であって, 本来の再帰的函数の定義とは違う. しかし, 函数が再帰的であることとミュー再帰的であることとが同値であるとわかっている.

問 69. 再帰的函数の理論について調べよ. □

標準型定理. (一変数の場合)[42] 特定の再帰的函数 Φ_1 があって, つぎのことがなりたつ. 任意の一変数再帰的函数 f に対応して自然数の定数 e が定まり, f の定義域の中のあらゆる x について

$$f(x)=\Phi_1(e,x)$$

がなりたつ. また, x が f の定義域にないときは, $\Phi_1(e,x)$ の値も存在しない. □

右辺 $\Phi_1(e,x)$ を f の標準型という. 標準型定理の証明は難しいので略すが, 概略の考え方はつぎのとおりである. Φ_1 は万能テューリング機械の働きを表す函数であり, 再帰的函数 f を計算するテューリング機械 M_f の構造をコード化した数が e である. 万能テューリング機械は e と x を与えられて M_f の動作を模倣して $f(x)$ を得る. この証明でテューリング機械の構造 (あるいは再帰的函数としての定義式) をコード化することが鍵であり, ゲーデル数の概念が重要な役割を果たす.

万能テューリング機械は, 標準型定理の Φ_1 を計算するテューリング機械である. プログラムををコード化してデータと同じに扱うというフォン・ノイマンのプログラム内蔵の考えは, 原理的に言えば万能テューリング機械を

[42] 二変数以上の函数についても同様であるが, 略す.

電子回路で実現しようということ, 言い替えれば Φ_1 を計算する機構を電子回路で作ろうということである. $f(x)$ を計算するには標準型 $\Phi_1(e,x)$ を計算すればよい. それには, Φ_1 を計算する万能テューリング機械に対して“プログラム”e と“データ”x とを与えればよい. これがコンピューターの原理であり, コンピューターが万能であるといわれる理由である. なお, テューリング機械のテープが無限に長いのにくらべて現実の機械は記憶容量などが有限であるという制約が避けられず, 原理的に計算可能であっても現実には計算できないということがあり得る. 実在のコンピューターは万能テューリング機械そのものというよりも, 万能テューリング機械の有限な近似であるとみるのが正確である.

いまコンピューター (電子計算機) とよばれているものは単に電子的な機構で計算する装置というだけでなく, フォン・ノイマン方式つまりプログラム内蔵方式のものの意味である. 電卓や ENIAC は, 電子的に計算をおこなうけれども, コンピューターではない.

コンピューターは計算可能な函数をすべて計算できるという意味で万能である. しかし, 計算可能でない函数も存在し, それはどんな機械によっても計算できない.

計算可能でない函数の例. まず函数 $\Phi_1(x,x)$ を考え, $f(x)$ を自然数 x についての場合分けで

$$f(x)=\begin{cases}\Phi_1(x,x)+1, & x \text{ が } \Phi_1(x,x) \text{ の定義域にあるとき}\\ 0, & x \text{ が } \Phi_1(x,x) \text{ の定義域にないとき}\end{cases}$$

と定めると, f は再帰的函数ではない. □

証明. 背理法の仮定として f が再帰的だとすると標準型定理により f に対応して定数 e が定まり, あらゆる自然数 x について

$$f(x)=\Phi_1(e,x) \tag{1}$$

がなりたつ. なぜなら, f はすべての自然数に対して定義されていて, 任意の自然数が f の定義域に属する. 式 (1) に $x=e$ を代入して

$$f(e)=\Phi_1(e,e), \tag{2}$$

したがって $\Phi_1(e,e)$ の値が存在し e は $\Phi_1(x,x)$ の定義域に属する. そこで f の定め方を考えると

$$f(e)=\Phi_1(e,e)+1. \tag{3}$$

式 (2) と式 (3) とはあきらかに矛盾する. □

計算不可能な問題としてはつぎの“停止問題”などがよく知られている. 上に述べた計算不可能な函数の例は停止問題の変形にすぎない.

停止問題. 与えられたテューリング機械とそのテューリング機械に与えられたテープに対して動作が終了して停止するかどうかを一般的に判定するテューリング機械は存在しない. □

一致問題. 与えられた二つのテューリング機械に同じテープをかけて起動したとき, 得られる結果が同じになるかどうか判定するテューリング機械は存在しない. □

問 70. 電卓とコンピューターとの違いについて説明せよ. □

問 71. 停止問題について調べよ. □

問 72. 計算不可能な問題のいろいろについて調べよ. □

計算量理論. 計算可能性の理論は 1930年代から研究されてきたが, 計算可能な問題でも計算が複雑で莫大な計算時間や記憶容量を要するものがあり, 実際的な観点から計算の“複雑さ”を考察する必要があることから, コンピューターの実用の広まってきた時代に計算量の理論が生まれた. 特定のコンピューターの能力に依存することなく一般論として計算がどのくらい複雑であるか考えるのが計算量理論 (complexity theory) である.

一例として, きわめて急激に増加する例として有名な

$$A(1) = 1 + 1, \quad A(2) = 2 \cdot 2, \quad A(3) = 3^3, \quad A(4) = 4^{4^{4^4}}, \ldots$$

となる函数[43]を考えてみる. これは二重再帰法で

$$\alpha(0, x, y) = x + 1, \quad \alpha(1, 0, y) = y, \quad \alpha(2, 0, y) = 0,$$
$$\alpha(z, 0, y) = 1 \quad (z \geq 3),$$
$$\alpha(z + 1, x + 1, y) = \alpha(z, \alpha(z + 1, x, y), y)$$

と定められた再帰的函数 α から

$$A(x) = \alpha(x, x, x)$$

と定められる再帰的函数であるから原理的には計算可能であるけれども, どんなにコンピューターの性能が高くなったとしても $A(4)$, $A(5)$, ... の値を求めることは現実的な時間内には不可能である.

問 73. $\alpha(1, x, y) = x + y$, $\alpha(2, x, y) = xy$ を確かめよ. □

問 74. $\alpha(3, x, y) = y^x$ を確かめよ. □

問 75. $A(4)$ の値のおおよその大きさを推定してみよ.[44] □

実用上も重要なさまざまな問題について, その複雑さが調べられている. 計算可能ではあるけれども, 工夫の如何にかかわらず計算はきわめて複雑であることわかっている問題も少なくない. 計算量理論の分野にはまたきわめて難しい問題として有名な $P = NP$ 問題をはじめ興味ある未解決の難問も多い.

問 76. $P = NP$ 問題とはどんな問題か. □

[43] W. Ackermann (1896–1962) が発見したもので, アッケルマン函数 (Ackermann function) とよばれる. ただしこれと似てはいるけれど別のペーテル函数 (Péter function) を混同してアッケルマン函数またはアッカーマン函数と書いている書物が多い.

[44] コンピューターで $A(4)$ の値を計算しようと試みてはいけない.

コンピューターの構造. テューリングの考えた万能テューリング機械をそのまま電子回路で実現しようとしても実用にふさわしくない. 万能テューリング機械はそもそも理論的考察を目的として実用を度外視して考えられたものである. 実際のコンピューターの構造は, 原理としては万能テューリング機械に基いてはいるが, つぎに述べるとおりフォン・ノイマンの提案に基いている.

コンピューターのハードウェアは, 中央処理装置 (CPU, central processing unit), 記憶装置, 入出力装置などの部分からなりたっている. 記憶装置 (memory) は主記憶と補助記憶 (外部記憶) とからなり, 主記憶は CPU から直接にデータのやりとりが高速にできる. 主記憶は 1 ビットを保持できる素子の集まりでバイト (8 ビット) ずつに区切ってあり[45], バイトごとに番地 (address) がつけてあって, 番地によってデータの所在を指定できる. 主記憶が IC (集積回路) などからなるのに対して, 補助記憶には磁気ディスクなど, 主記憶とくらべて遅くても容量の大きい媒体がもちいられる.

問 77. 集積回路について調べよ. □

問 78. 記憶媒体のいろいろについて調べよ. □

CPU は置数器 (register), 制御装置, 演算装置などからなる. プログラムは命令の列としてコード化して主記憶に納められる. 制御装置は主記憶から命令を一つずつ取り出して解釈し実行する. 命令に従って制御装置は演算装置を駆動したり, 主記憶と置数器との間でデータのやりとりをしたり, 補助記憶装置や入出力装置を動かしたり, つぎにおこなう命令を選択したりする. すなわちプログラムによる逐次制御がおこなわれる. 演算装置は制御装置からの指令に従い主記憶からデータを取りだして置数器に置いてあるデータとの間で演算をおこなったりする.

CPU はコンピューターの中でも構造の最も複雑な部分であり, トランジスタなどの素子を多くつなぎ合わせて作られてきたが, 現在では CPU を一個の集積回路で作った MPU (MicroProcessor Unit) が普及していて, パソコンに使われるのみならずもろもろの電気機器に組み込まれて制御用に使われるなど, 世の中に大きく影響している.

問 79. MPU の構造について調べよ. □

問 80. MPU の歴史について調べよ. □

問 81. MPU の応用例について調べよ. □

入出力装置はキーボード, マウス, ディスプレイ, プリンター, 通信回線などコンピューターと外部とのデータのやりとりを受け持つ.

問 82. ハードウェアの構造について調べよ. □

現代のソフトウェアは, ワープロソフトや表計算ソフトや WWW ブラウザーやエディターやコンパイラーなど個々の応用のためのアプリケーションソフトと, それらのソフトを管理し制御する OS (operating system) などからなる. さらに入出力や通信などの制御も直接におこなうのは非常に難しい

[45] 昔のコンピューターの多くは十数ビット〜数十ビットの“語” (word) に区切っていた.

のでそれらの操作は OS がおこなっていて, 一般利用者は OS を介して入出力などをおこなう.

問 83. OS について調べよ. □

コンピューターの仕事をハードウェアとソフトウェアとでいかに分担するかということは必ずしも一定していない. たとえば数値の演算について, 加減算などの簡単なものは前述の論理回路をもちいてハードウェアでおこない, 平方根や三角函数などはソフトウェアでおこなうのがふつうであるが, 除算などをハードでおこなう場合もソフトおこなう場合もある.

現代のコンピューターは, プログラムもデータもビット列としてコード化することによって, プログラムをもデータとして扱い得ることになり, 万能テューリング機械の原理に基いて万能の力をもっている. さらに, コード化する考えで数値以外のデータを処理する非数値計算 (nonnumerical computation) もおこなうことができる. 実際, 現代のコンピューターは文書, 画像, 音声など数値以外のもろもろのデータを扱うことができる. “計算機”という呼び名にもかかわらず, いまは数値計算以外の処理の方がむしろ多くなっている. 従って“計算機”とよばれていても実態は汎用情報処理機である.

問 84. コンピューターで数値計算以外にどんなことができるか調べよ. □

参考文献

注. 著者名五十音順. 講義の内容に直接に関係するものに限らず, 計算可能性の理論に関するものなど参考になりそうなものをあげておく. なお, いま 売られているか否か 確かめてない. 品切れまたは絶版の場合もあり得る.

廣瀬 健: **数学的帰納法**. シリーズ新しい数学の応用 11, 教育出版, 1975.

町田 元, 横森 貴: **計算機数学**. 森北出版, 1990.

小野 寛晰: **情報科学における論理**. 日本評論社, 1994.

小野 寛晰: **情報代数**. 情報数学講座 2, 共立, 1994.

高橋 正子: **計算論 ——計算可能性とラムダ計算——** コンピュータサイエンス大学講座 24, 近代科学社, 1991.

上に挙げた参考文献のほか

岩波数学辞典 (日本数学会編, 岩波書店, 1985)

数学小辞典 (矢野健太郎・茂木勇・石原繁編, 共立出版, 1968)

情報処理ハンドブック (情報処理学会編, オーム社)

など辞典・事典類をも参照. 歴史的に計算の理論とかかわりの深い**ゲーデルの不完全性定理**については林 晋 (すすむ) (神戸大学) のウェブサイト
`http://kurt.cla.kobe-u.ac.jp/~hayashi/`参照.

[終]