アルゴリズム概論 I

野崎昭弘†

本解説の目標　アルゴリズムという言葉は，情報処理の基礎用語のひとつであり，今さら言葉の解説をすることなど，あまり意味がないのではないか，とも思われる．しかしながら，手近の辞書で調べてみると，この言葉の正確な記述はあまりみあたらない．(広辞苑には，オートマトンがのっているのにアルゴリズムはのっていない．研究社新英和大辞典には，algorithm：互除法と記されている．)したがって，この言葉の平易かつ標準的解説をしておくことには，資料的価値があるように思われるし，またアルゴリズムについての（最近のものも含めて）数学的理論の展望をしておくことも，無益とはいえないようである．

以上の観点から，ここでは予備知識をまったく仮定せずに，白紙の状態から言葉の解説を始めることにした．それは冗長であるかも知れないが，後々考えられる問題を，共通の理解のもとで整理するためには，欠くことができないことと思われる．その後で，計算可能性や計算量の理論について，簡単な展望と基本概念の考察を試みたいと思う．これがアルゴリズムの実際的に問題にたずさわっておられる方々にも，何かの参考になるところがあればと願っている．

1. アルゴリズムの講義

アルゴリズムとは，平たくいえば，情報処理の『しかた，手順』を意味する言葉である．語源的には，アラビアの数学者 Al-Khwarizmi (780 頃—850 頃) の名に由来し，古くは（彼が採用した，0 を含む）アラビア式十進記数法を意味する言葉で，algorism と綴られた．その後，意味が少しずつ広がってアラビア数字による計算法や，アラビア数字そのものをも意味するようになった．(この後の用法から，“a cipher in algorism：ゼロ，有名無実の人”のような表現が生まれた．) しかも計算法として，単なる加減乗除だけでなく，最大公約数を求めるためのユークリッドの互除法をも含むようになったが，その過程でギリシャ語の『数』arithmos と混同されて，algorithm と綴るようになった．現在では，いわゆる計算だけでなく，文字処理をも含むディジタル情報処理一般の『手順』をアルゴリズム (algorithm) と呼ぶのがふつうである．

アルゴリズムに似た言葉として，プログラム (programme) がある．プログラムには『計画』という抽象的な意味もあるので，計算機のプログラムの場合には，アルゴリズムと同じような意味で使われることもある．しかしながら，プログラムには，計画を書きあらわした『もの』(たとえば曲目の演奏順序を印刷を印刷したパンフレット) という意味もあるので，注意が要る．幸いアルゴリズムには，抽象的な『手順』という意味しかないので，区別をはっきりさせるためには“このアルゴリズム（方法）を COBOL 語で書きあらわしたプログラム (program, もの)”のような表現が便利である．

アルゴリズムの概念は，計算機による情報処理の発達とともに，実用上も注目されるようになった．(理論的には，数理論理学の『証明可能性』の概念との関連において，いち早くとりあげられていた．) たとえばなんらかの基準で，資材の最適な配分法を考える場合には，『うまい配分法が存在する』というのもひとつの答であるが，『このように配分すればよい』という答の方が望ましいことは，いうまでもない．それゆえ，資材についてのデータから最適な配分法を求める，アルゴリズムが必要とされるわけである．

ここで問題になるのは，次の 3 点である．

（1） 客観性——機械的に追跡できる行動の規則として，客観的に記述できるか？

（2） 有限性——有限時間内に，有限の材料（たとえば計算用紙）によって，実行可能であるか？

（3） 正当性——求められた答が，正しいという保証があるか？

計算機が誕生した頃，夢多きパイオニアたちの中には，これらの条件をみたすアルゴリズムがみつかりさえすれば，どんな問題でも解けるかのように考えた人が多かった．それは文字通りの意味では正しくなかっ

† 東京大学理学部情報科学研究施設

たけれども，『現実的に実行可能』ということの意味が，計算機の登場によって大幅に変えられたことはたしかである．(たとえばキングサイズの連立1次方程式を解く，ロケットの（半）自動制御など．)

次にこれらの性質について，もう少し詳しく観察して，問題点をピックアップしておこう．

2. アルゴリズムの客観性

あるアルゴリズムが客観的であるというためには，少くとも，それが他人にわかる形で記述されなければなるまい．当然，客観性の問題は，記述の方法，すなわち言語の問題と関係をもってくる．

言語はおおざっぱにいって，単語（の集団）と文法規則（の体系）から構成される．アルゴリズムを記述する言語の場合には，まず単語を組みあわせて基本操作をあらわす文を作り，その文を組みあわせて（文章プログラム）を作るのがふつうである．したがって客観性の問題は（有限性・正当性もそうであるが），単語レベルの問題と，文レベルの問題，それに文の組み合わせかたの問題に還元される．このうち単語レベルの問題が基本的であるから，最初にとりあげてみよう．

アルゴリズムを記述するひとつひとつの文には，ひとつの基本処理操作あるいは判断操作をあらわすひとつの動詞（句）と，処理の対象となる個個の物をあらわすいくつかの名詞 (name, identifier) が含まれていると考えてよい．使われる動詞の種類はあらかじめきめられているのがふつうであるが，どのようなものがどれだけ選ばれているかは，アルゴリズムの吟味をするときには致命的な問題である．あとでの議論にも関係するので，いくつかの例を挙げておこう．

（A） ユークリッドの互除法を実行するために：

処理操作 (a 1) m を n で割って，余り（非負最小剰余）r を求める．

(a 2) 数 x の値を，(あらためて) y とおく．

(a 2) は，図1では記号的に

$$y \rightarrow x$$

とあらわされている．

判断操作 (a′1) $m \geqq n$ か？

(a′2) $r=0$ か？

（B） 初等幾何学の作図アルゴリズムでは：

処理操作 (b 1) 指定された点を通る直線をひく．

(b 2) 指定された点を中心とし，指定された半径をもつ円を描く．

(b 3) 指定された2直線の交点を求める．

(b 4) 指定された2円の交点を求める．

(b 5) 指定された円と直線との交点を求める．

判断操作 (b′1) 指定された点が，指定された直線，半直線，線分，円あるいは円弧の上にあるか？

(b′2) 指定された直線，半直線，円あるいは円弧どうしの，交点が存在するか？

（c） FORTRAN 語では：

処理操作——代入文，READ 文，WRITE 文．

判断操作——算術 IF 文，論理 IF 文，DO 文など．

使われている変数の意味を確定することは，宣言文 (COBOL では，データ部) で行なわれる．実用上は，

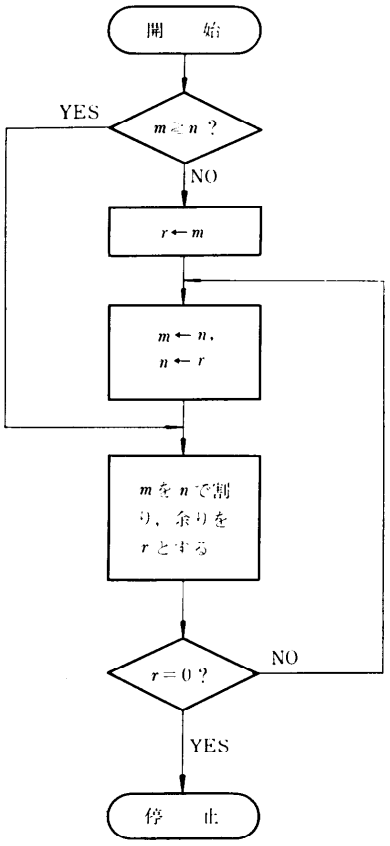

図 1 m, n の最大公約数を求めるアルゴリズム．答は n に求まる．(入出力の部分は省略して計算の部分だけ示した．) なお，$m, n>0$ と仮定する．

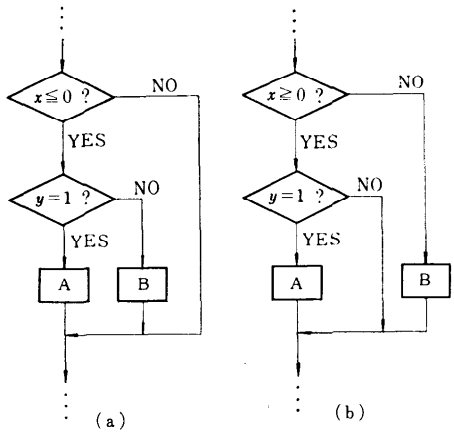

図 2

この部分に処理上むずかしい点が多く，客観的な解釈を保証するための正確な定義が強く要求されるところである．

これらの文をつなぎ，全体の流れを指定することは，流れ図のような図式言語では矢印によって示し，FORTRAN 語などでは制御文（GO TO 文など）によって行なわれる．このレベルでは，記述された（複合）文の『構文的一義性』が，客観性の問題として重要である．簡単な例を挙げるならば，次のようにアイマイな文は困るのである．

“もし $x \geqq 0$ ならば，$y=1$ のときに A を実行し，
さもなければ B を実行せよ”

これは，あらかじめ解釈を統一しておかないと，図2（a），（b）のような2通りの解釈が可能になり，客観性が損なわれる．（かつて Algol 語の if 文の解釈で問題になったことである．）

3. アルゴリズムの有限性

ある問題を解くための手続きが，いかに客観的に正当なものであっても，それが無限の時間あるいは無限の資材を前提とするものであっては，実行可能とはいえない．また，数学的に有限で，実行可能（これを effective という）であっても，その有限の大きさの程度によっては，たとえば全宇宙の素粒子と同じ個数のディスク・パックを必要とするような計算は，事実上実行不可能である．

ここでふたつの問題が生ずる．

（1）　与えられた問題を解くための『有限的』手続きが存在するか？

（2）　あるアルゴリズムを実行するために，どの程度の時間および資材が必要とされるか？

前者は理論的な問題であるが，否定的に解かれた場合には，（警告という）実用上の意味があると考えられるし，また肯定的に解かれた場合にも，その解かれかたによっては（特によいアルゴリズムが具体的に示された場合）実用上も大きな意味をもちうる．後者は実際的な問題であるが，くわしく調べてゆくと，数学的にも興味深い未解決の問題が，数多く見出される．

ところで，前章で述べた『動詞の選びかた』（基本操作の選びかた）は，これらの問題のどちらにも大きく影響する．たとえば有名な『与えられた角の3等分』の作図にしても，基本操作を前章の（B），すなわち定規とコンパスでできる範囲に限れば，一般の作図アルゴリズムは存在しない（特殊な場合なら，できることもある――たとえば直角の3等分）が，直交座標軸と $\tan\theta$ のグラフが与えられたとして，基本操作としてそのグラフと指定された直線との交点を求める作業をゆるすならば，任意の角の3等分が可能になる．

ユークリッドの互除法において，基本操作として割り算（a 1）を許さず，代りに引き算

（a 3）m から n を引いて，差を求める

を許すことにしたらどうであろうか？その場合には，流れ図をいくらか修正しさえすれば，やはり最大公約数を求めることができる（図3）．すなわち可能性の問題（1）には影響を及ぼさない．しかし，能率の問題（2）には関係してくるので，たとえば

$m=2744$,　　$n=226$

の場合，図1のアルゴリズムで解けばループを2回ま

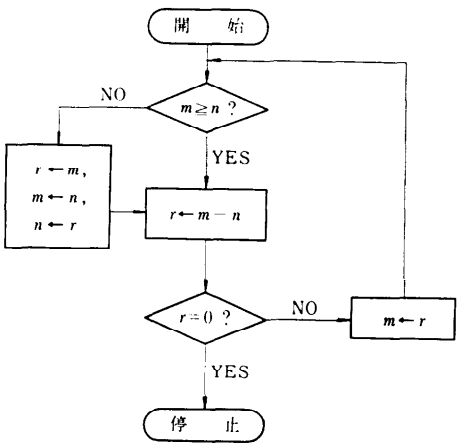

図 3　減算を基本にした，互除法

わるだけで答が求まるが，図3のアルゴリズムでは，ループを 34 回まわらなければならない．ループ1回あたりの所要時間が異るからこれだけでは正確な比較はできないが，少くとも『計算時間が大きく変わりうる』ことはこの例からもおわかり頂けるであろう．

能率の問題を正確に議論するためには，時間や資材の消費量を測る単位をきめておかなければならない．それにはいくつかの立場があるが，ある種の計算機を想定して，実行される命令のステップ数と，消費される記憶装置の語数とを測る立場と，加減乗除の演算を1単位として，演算実行回数を評価する立場とが代表的である．実用上は，外部記憶装置の使い方や，入出力機器との通信をも含めた，データ転送の際の待ち時間の評価が重要であるが，これらについての一般論は困難なので，ふつうはシステム評価の問題として個別的にとり扱われるのが現状である．

4. アルゴリズムの正当性

アルゴリズムが正しいか正しくないかを議論するためには，まず問題それ自身が正確に定式化されていなければならないし，またその問題の解が，そのアルゴリズム（具体的には，プログラム）の中で，どのように表現されているかがはっきりしていなければならない．たとえば，ユークリッドの互除法（図1）の場合には，正しい最大公約数を d としたとき，この d の値が最終的に n のところに求まることが意図されている．いいかえれば，プログラムが停止した時点で，命題

"$n=d$"

が成りたつはずである．このような，『成りたつべき命題』が指定されてはじめて，正当性の意味が確定すると考えられる．上の例では，$r=0$ のときにだけプログラムが停止することと，m, n の最大公約数と n, r（m を n で割った剰余）の最大公約数とが一致することから，停止した時点で $n=d$ となることが簡単に証明される．

正当性にかかわる問題は重要なものがたくさんある．思いつく順に，個条書きにしてみよう．

（1） 答が実数であらわされる場合の正当性――この場合には，答を正確に求めることが，有限時間内には原理的に不可能であろう．そこで次のふたつの立場のどちらかがとられることになる．

（a） どのように詳しい値でも，有限時間内に求めることができるなら，それでよいと考える――たとえば『円周率 π の値を（Turing の意味で）計算するアルゴリズムが存在する』というのは，この意味である．たとえ何億桁であろうと，充分な時間と記憶装置がありさえすれば，正しく求めることができる．

（b） 正しい値との差（誤差）が，ある指定された限界より小さくなればよいと考える――代数方程式の根を求めるプログラムが，精度なにがしの範囲で正しい，などというときは，この意味である．

（a），（b）にはもちろん密接な関係があるが，計算の可能性などを一般的に考えるときには（a）の立場を，個々のプログラムの正当性を問題にするときには（b）の立場をとることが多い．答が**複素数**，あるいは実関数など，要するに**離散的でない量**が対象になる場合には，これと同じ問題が発生するのはもちろんのことである．

（2） 成りたつべき命題の記述法――整数値関数を計算するプログラムの場合には，算術式間の等式を含む，いわゆる述語論理式によって，成りたつべき命題が比較的容易に表現されるであろう．しかしながら，たとえばコンパイラや一般の事務処理のように，記号列の処理過程については，成りたつべき命題を記述することがすでに（よほどうまい記述法が工夫されない限り）困難になることと思われる．

（3） 正当性の判定の（半）機械化――成りたつべき命題がはっきりすれば，それが成りたつかどうかをなるべく機械的に検査することは，重要な問題である．あとで述べるように，広範囲な問題について正当性を判定するアルゴリズムは存在しないので，理想的な機械化は不可能であることがわかっているが，問題の範囲を制限することによって，正当性の機械的判定を可能にすること，またそれが可能である場合に，その判定アルゴリズムの能率を吟味すること，また一般の問題について，判定作業を部分的に機械化し，人間の負担を軽くすることなどが研究されている．

（4） 停止性の問題――互除法の例では，プログラムが停止した時点で命題 "$n=d$" が成りたつことを考えた．そこで当然，『プログラムが本当に停止するか？』という，停止性の問題が派生してくるわけである．

プログラムが停止することをも，成りたつべき命題の中に，必要条件として書き加えておくことは（記法上の制約さえなければ）可能である．また逆に，プログラムを修正して，正しい答が求まったときにだけ停止するようになおすことができるから，正当性の問題

と停止性の問題とは等価である．しかし，どのような表現をえらぶにしても，問題のむずかしさが軽減されるわけではなく，停止性を判定する一般的なアルゴリズムは存在しない．そこで，ある限られた範囲での停止性判定アルゴリズムが考察されることになる．

（5） 等価性の問題——ふたつのプログラムが，同じ答を出すという意味で等価であるかどうかは，実用上しばしば問題になる．たとえば計算機システムを更新するとき，旧ライブラリを新しいシステムにあわせて書きなおすことが必要になるが，そのとき『誤りを犯さない』とは『等価性を保つ』ことに他ならない．また，プログラム A の能率向上を図って，プログラム B を作ったとき（optimize 等）にも，A と B との等価性が問題になる．プログラム A が正しいと仮定すれば，A と B との等価性をたしかめることは B の正当性をたしかめることにもなる．

等価性の判定も，一般的には不可能である．したがって，(3)，(4) と同様に，限られた場合についての判定アルゴリズムや，一般的発見的 (heuristic) システムがとりあげられるわけであるが，そのどちらについても，アルゴリズムやそれがみたすべき条件の『記述法』が大きな問題になることを注意しておこう．

さて，以上の観察を通じて，次のような問題がひろい出された．

（1） 客観性について——1.1 文法の問題
　　1.2 解釈の問題
（2） 有限性について——2.1 アルゴリズムの存在
　　2.2 アルゴリズムの評価
（3） 正当性について——3.1 誤差の評価
　　3.2 正当性の表現
　　3.3 停止性
　　3.4 等価性

これらのうち(1)は，言語の問題であって，アルゴリズム論の問題というよりは，さらに深く広い問題であるともいえる．(2)以下がアルゴリズム論独自の問題といえるので，次にこれらに関係する理論を，その基本概念の解説を中心に，概観してゆきたい．

5. アルゴリズムの一般論

アルゴリズムが『存在』することを示すには，ひとつ実例を書いてみせるのが早道であろう．そのときにはアルゴリズムの一般的な定義などなくても，その実例さえ充分明瞭に記述されれば，存在証明として説得力をもつであろう．しかし，アルゴリズムが『存在しない』ことを示すときにはそうはいかない．ありとあらゆるアルゴリズムを含みうるように充分一般的で，しかも『存在しない』ことの証明ができるように充分厳密な，アルゴリズムの定義をまず与えておかなければならない．それにはいくつかの方法があるが，チューリング (A. M. Turing, 1912—1954) のやり方が直観的に最もわかりやすいと思われるので，彼の機械の紹介から始めることにしたい．

チューリングは，アルゴリズムの客観性を保証する最も確実な手段として，機械による情報処理を考えた．扱われる情報はディジタル型（離散型）であるとし，また機械の動作もディジタル的に，逐次的に行なわれると考えたので，彼が想定した機械は，現在の磁気テープつき計算機にきわめてよく似ている．

彼の機械——チューリング機械は，図4のように，本体と，ます目に仕切られた1本のテープ，それに1個の読み書きヘッドから成っていて，次のような動作を行なう．

(1°) ヘッドによって，その下のます目にある情報 x を読みとり，本体に送る．

(2°) 本体は，その時の状態 q と，読みとられた状態 x とに応じて，状態を q' に変え，ヘッドによってその下のます目に情報 y を書きこみ（x を y に修正する——$y=x$ でもよい），さらに次の動作のひとつを選択する．

(イ) テープを1ます目分右に送り，(1°)に戻る．
(ロ) テープを1ます目分左に送り，(1°)に戻る．
(ハ) 停止する．

さらに詳しい証明はすぐ後に述べるが，このような

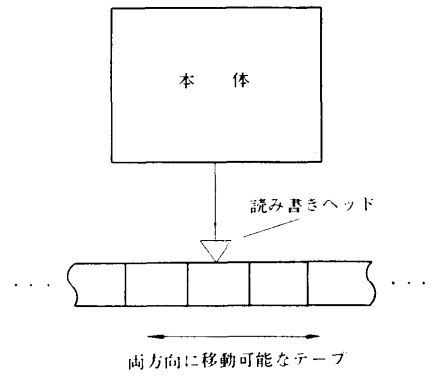

図 4　チューリング機械の概念図

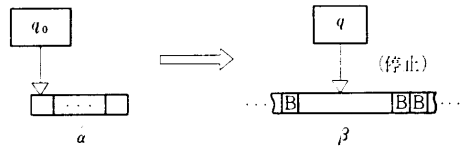

図 5 $\beta=\varphi(\alpha)$ を計算するチューリング機械の動作．なお便宜上，β の両側にあるBは無視してもよいものとする．

機械を想定することによって，情報処理アルゴリズムが次のように定義されるのである．

まず，情報を形式的に記号列としてとらえ，情報処理とは情報 $\alpha,\beta,\gamma,\cdots$ に情報 $\varphi(\alpha),\varphi(\beta),\varphi(\gamma),\cdots$ を対応させる関数 φ のことである，と考える．すると，情報処理 φ を実行する（チューリングの意味の）アルゴリズムが，次のように定義される．

定義 情報処理 φ を実行するアルゴリズムが存在するとは，適当なチューリング機械 T とその状態 q_0 が存在して，次の条件をみたすことをいう．

『機械 T に，情報 α（または $\beta,\gamma,\cdots$）を記入したテープをかけ，ヘッドをその左端において，状態 q_0 から動作を開始させると，有限時間内に必ず停止して，しかも停止した時のテープには，$\varphi(\alpha)$（または $\varphi(\beta),\varphi(\gamma),\cdots$）が記入されている（図5）．』

このとき，関数 φ は（チューリングの意味で）**計算可能**であるともいう．やかましくいえば，φ の定義の前に，記号の有限集合 Σ をひとつきめておき，Σ の中の記号の，すべての有限列の集合を Σ^* とした上で，“φ は Σ^* から Σ^* への部分関数である（Σ^* 全体で定義されていなくてもよい）”といわなければならない．

さて，このような機械の有限性を保証するために，チューリングは次のことを仮定した．どれも，ふつうの電子計算機の場合にも当然と考えられる条件である．

（1） 本体のとりうる状態は有限個しかない．

（2） テープのひとつのます目に書きうる情報は，有限個しかない．

チューリングは，有限個しかない情報 $x_1,\cdots,x_N$ のひとつひとつを，あっさり**文字**と呼び，ひとつのます目には1文字しか書けないことにした．また文字の中には，**空白記号**（以下 B であらわす）と呼ばれる，“また何も書かれていない，白紙の状態”をあらわすものがきめられているとした．これは，一般性を獲得するための次の仮定に関係している．

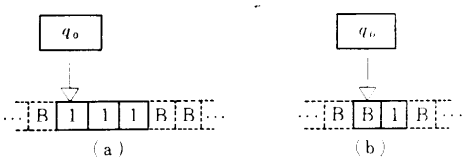

図 6

（a） ヘッドは右に動き，1をひとつつけ加えて停止する．

（b） テープが右へ右へと送られ続け，永久に停止しない．

表 1

q	x	q'	y	その後の動作
q_0	B	q_1	1	右
q_0	1	q_0	1	左
q_1	B	q_1	B	右
q_1	1	q_1	1	停 止

状態 q で文字 x を読んだときの，機械 T_0 の動作をあらわす．q' は次の状態，y は書きこまれる文字である．右，左とあるのは，テープを移動する方向である．（ヘッドは相対的に，逆方向に動くことになる．）

（3） テープの長さは有限であるが，必要に応じて（ヘッドがテープの外に飛びだしそうになったとき）いくらでも，空白記号 B が書きこまれたます目をつぎたすことができる．

これはわれわれの計算機でいえば，磁気テープを必要に応じていくらでもつぎ足してやれることに相当する．これを仮定しておかないと，電子計算機には加算もできない（$\pi+e$ の，(100!) 桁計算は，事実上不可能であろう）ことになるので，アルゴリズムの一般論を考えるときには不適当なのである．

チューリング機械のひとつの例を，表1に示しておいた．この機械 T_0 の本体のとりうる状態は q_0,q_1 のどちらかしかなく，ひとつのます目に書きうる文字は，1 と B しかない．この T_0 を，図6のような初期状態から出発させるとどのような動作をするか，ひとつ考えてみて頂きたい．

ところで，このような機械でできることに，どれくらいの一般性があるのか，疑義をもたれる方もあることと思う．しかしチューリングの機械は実によく考えられていて，およそ客観的・有限的と考える手順はどれも，適当なチューリング機械に実行させることができると思われる．このことは，『手順』の一般的な定義が与えられない限り，『証明』することはできないが，それを信ずるに足ると思われる理由をいくつか挙

げてみよう.

まず第1は，現在のどんな電子計算機も，一種のテューリング機械と考えることができる，という点である．磁気テープやディスク・パックのように，交換可能な部分を『テープ』と考え，それ以外の部分（コアメモリ，磁気ドラムなどを含む）をすべて本体と考えれば，ありうる状態の数が莫大になるとはいえ，たしかに（ディジタル型である以上）有限だから，テュリーングの仮定に反するところはない．ただ，『テープは1本，ヘッドは1個』というところが難点であるが，これは『多テープ・多ヘッド』テューリング機械というものを考えれば，解決がつく．そして『多テープ・多ヘッド』テューリング機械にできるどのような作業も，必ず適当な（前に定義した1テープ・1ヘッド）テューリング機械によって simulate できることが証明されるのである．したがって，われわれの関心が『電子計算機によって，原理的に解けるかどうか』であるならば，テューリングの機械は充分一般的なモデルであるといえる．

第2の理由は，人間が行なうディジタル情報の処理過程を観察して得られる．処理が客観的に行なわれるとすれば，人間は頭の中に記憶されている処理基準や途中経過に応じ，また計算用紙に記録されているデータに基いて，機械的に作業を進めることであろう．そこで次のことを仮定しても，一般性を失わないと考えられる．

（1） 計算用紙は幅の広いテープであって，ページに仕切られている．1ページには，あるきまった個数より多い記号は書けない．（記号の種類も有限個とする．）

（2） 計算用紙を見るときに，ひと眼で見られるページ数は限られている．

（3） 計算用紙の他の場所をさがすときには，次々と隣のページを見てゆくか，あるいは目印に“クリップ”をはさんでおいたところに飛ぶことしかできない．（ある超越的方法で，都合のよいページがぱっとみつかるようなことは，許さない．）

また，有限性を守るために，次の仮定を設けることは当然であろう．

（4） 頭の中のありうる状態は有限個しかなく，また同じ状態で同じ情報を読めば，同じ動作を行なう，その動作とは，頭の中の状態をかえ，見ているページの情報を書きかえ，また見ている位置と，クリップの位置を変更するか，あるいは作業を停止する．

（5） クリップの個数は有限個（一定個数以下）である．

（6） 計算用紙がなくなったときには，白紙のページがつぎ足される．（カンニングは許さない！）

これらの仮定の下では，人間の動作を（多ヘッド）テューリング機械の動作とみなすことができる．それには頭脳を本体とみなし，計算用紙をテープとみなし，また『ひと眼で見られるページ数』をひとつのます目と考えればよい．するとひとつのます目に，多数の記号が書きこまれることになるが，その組合せ（順列）は有限個であるから，テューリングの意味の『文字』はやはり有限個におさえられる．そして，クリップのある位置に，ヘッドがあるものと考えればよい．

こうして得られる，拡張された『多ヘッド』テューリング機械が，次の機能をもったとしよう．

（a） どのヘッドも，そのま下と両隣の，3個のます目の情報を読み書きできる．

（b） ヘッド i を，左右に1ます目分ずらしたり，またヘッド j の位置にいっぺんに移すことができる．

そのようにしても，ヘッドが一定個数以下で，動作が本体と読まれた情報とによって決定論的に（同じ規則で）行なわれるものなら，その動作を simulate する（1ヘッド）テューリング機械をつくることができる．このように，テューリング機械は融通のきくものなのである．

第3の理由は，これまでの経験である．アルゴリズムを充分一般的に定義しようという試みは，他にもいろいろあるが，その中で最も一般的なもの（Kleene, Church, Markov, Smullyan 等によって，見かけ上全く異るものが独立に与えられた）が，どれもテューリングによる定義と同等であることが証明されるのである．この事実は，これらの定義が充分な一般性をもつと信じるための，有力な傍証を与えてくれると考えられる．

さて，テューリング機械について，次の事実が基本的である．

事実 1 ひとつのテューリング機械は，ひとつの（表1のような）表によって，有限的に記述される．表の各行を横につなげれば，ひとつの記号列として記述されるといってよい．

事実 2 テューリング機械の状態や文字を，いつでも

$q, q', q'', \cdots$ や $\mathrm{B}, \mathrm{B}', \mathrm{B}'', \cdots$

であらわすことにしよう．すると，記号

q, B, $'$ (ダッシュ), 右, 左, 停止

をそれぞれ

000, 001, 010, 011, 100, 101

であらわすことにすれば，勝手な状態，文字，文字列，また勝手なテューリング機械が，記号0,1の列としてあらわされる．

事実 3 テューリング機械の停止問題を判定するアルゴリズムは，存在しない（停止問題の非可解性）．

このことは，少し詳しくいえば，次のことを意味している．まず，停止問題の判定を，情報処理として次のように定式化する．

$$\varphi(\alpha)=\begin{cases}1\cdots\cdots\alpha \text{ が以下の条件 } (*)\\ \qquad\text{をみたすとき}\\ 0\cdots\cdots\text{それ以外のとき}\end{cases}$$

ただし α は，0,1 の列であって，(*)とは，

(*1) $\alpha=\beta 111\gamma 111\delta$ という形をしている．

(*2) β は，あるテューリング機械 T を記述する記号列である．

(*3) γ は，T のある状態 q_0 をあらわす記号列である．

(*4) δ は，T のあるテープ μ をあらわす記号列である．

(*5) T にテープ μ をかけ，その左端にヘッドにおいて状態 q_0 から出発させると，有限時間内に停止する．

すると，次の定理が成りたつのである．

定理 1 情報処理 φ を実行するアルゴリズムは存在しない．（関数 φ は，計算可能でない．）

〔略証〕 もし φ があるテューリング機械 T によって計算可能なら，T をわずかに手なおしすることによって，次のような機械 T_0 がつくれる：T_0 にテープ β をかけて，初期状態 q_0 で出発させると，

T_0 が停止する $\Longleftrightarrow \varphi(\beta 111\gamma_0 111\beta)=0$

ただし γ_0 は q_0 をあらわす記号列である．そこで

$\beta=T_0$ をあらわす記号列 β_0

の場合を考えると，

T_0 が停止する $\Longleftrightarrow \varphi(\beta_0 111\gamma_0 111\beta_0)=0$

$\Longleftrightarrow T_0$ が停止しない．

これは矛盾である． 〔証終〕

これはアルゴリズムの一般論の限界を示す．重要な定理である．この定理から，プログラムの正当性・等価性を判定する一般的アルゴリズムが存在しないことも，ただちに導かれるわけである．

計算量および正当性に関する理論については，次回に述べることにしたい．

（昭和 48 年 9 月 3 日受付）

資　　料

PDP-11/20 デュアルシステムにおけるデバイスシェアリングシステムの製作†

本田　直人††　葛山　善基††　藤井　護††　都倉　信樹††

Abstract

A PDP-11 dual system with an interprocessor communication channel (DA11-D) has been installed at Osaka University. Development work started on a device-sharing system, in which a user can use all devices connected with another CPU as if they were of its own.

This paper describes the design and implementation of a device-sharing system which is consistent with a manufacturer-supplied disk operating system, and discusses some of the problems encountered.

This system began the operation in December 1972 and is now used successfully.

1. はじめに

近年，複数の CPU（中央処理装置）を持ついわゆるマルチ・コンピュータ・システムが増加している．それにともないミニコンのネットワークに関する研究も各地で盛んに行われるようになった[1]．大阪大学基礎工学部情報工学科には，2台のミニコン PDP-11/20 とこれらを結ぶインタフェース装置（DA11）から成る図1に示すようなデュアルシステムがある．これらA・B両システムは，導入時にはそれぞれ DOS(Disk Operating System)のもとで独立に動作できるだけで，Aシステムからは高速紙テープリーダなどを，またBシステムからはラインプリンタなどを使用できなかった．もし，このようなデバイス（入出力装置）を A・B両システムから使えば，各システムにデバイスを増設したのと同じ効果を得ることができる．そこで既存の DOS を改造して，コンソールとテレタイプを除くすべてのデバイスを両システムから使えるようなデバイスシェアリングシステムを開発した．以下，2章では PDP-11 の概要を紹介し，3章でこのシステムの作製の基本方針を，4章ではここで用いたデータ転送

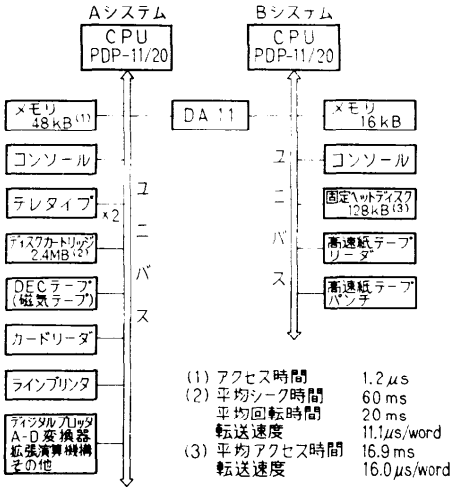

図 1　デュアルシステムの構成
Fig. 1　The dual system configuration

の方法を，5章ではロック・アンロックの問題を，また6章ではシステム全体のデバイスを管理するデバイス管理ルーチンについて述べる．

2. 既存システムの概要

2.1 ハードウェア[2]

○ユニバスによる共通母線方式を用いている．

○各デバイスの制御装置内のレジスタは，メモリと

† A device-sharing system for a PDP-11/20 dual system, by Naoto HONDA, Yoshiki KATSUYAMA, Mamoru FUJII and Nobuki TOKURA (Faculty of Engineering Science, Osaka University)

†† 大阪大学基礎工学部情報工学科