■6 群（コンピュータ - 基礎理論とハードウェア）- 2 編（計算論とオートマトン）

4 章 チューリング機械

（執筆者：岩本宙造）［2010 年 3 月 受領］

■概要■

本章では，チューリング機械，句構造文法，万能チューリング機械，停止問題の決定不能性，帰納的可算，チャーチ・チューリングの定立について概覧する．まず，計算機の数学的モデルのなかで，最も重要なものであるチューリング機械について，その概念や定義，性質，いくつかのバリエーションモデルを紹介する．次に，チョムスキーによって考案された文法体系である句構造文法について，チョムスキー階層やチューリング機械との関係などを説明する．また，あらゆるチューリング機械を，それ一台でシミュレートできる万能チューリング機械について，その構成法や，小サイズ化に関する最近の研究を紹介する．さらに，与えられたアルゴリズムが，与えられたデータに対して停止するか否かを決定する停止問題と，その問題の決定不能性について概説し，最後に，チャーチ・チューリングの定立について述べる．

【本章の構成】

本章は，チューリング機械（4-1 節），句構造文法（4-2 節），万能チューリング機械（4-3 節），停止問題と決定不能性（4-4 節），帰納的可算（4-5 節），チャーチ・チューリングの定立（4-6 節）の 6 節からなる．

■6 群 - 2 編 - 4 章

4-1 チューリング機械

（執筆者：森田憲一）[2009 年 1 月 受領]

チューリング機械は 1936 年にチューリング（A.M. Turing）によって考案された計算機の数学的モデルであり，種々の計算機モデルのなかでも第一にあげられるべきものである．本節では，チューリング機械がどのようにして考案されたかを解説し，標準的なモデルの定義と例を与えた後，いくつかのバリエーションについて説明する．

4-1-1 チューリング機械とは

チューリングは 1936 年の論文 [7] のなかで，計算者（computer）である人が紙，鉛筆，消しゴムなどを使って行う計算の過程を思い浮かべて分析し（図 4･1 はその想像図），その動作を非常に単純な，思考上の機械によってモデル化した（なお，ポスト（E.L. Post）もこれと同等のモデルをチューリングとは独立に同年に考案している[6]）．

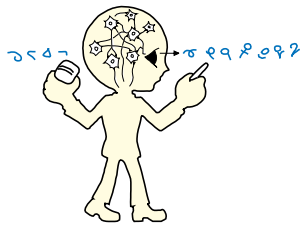

図 4･1　計算者による計算の様子

チューリング機械は，概念的には図 4･2 のように，ます目に分割され左右に無限に伸びたテープ，有限制御部，及びテープ上の記号を読み書きするためのヘッドから構成される．チューリング機械は，$0, 1, 2, \cdots$ といった離散的な時刻ごとに次の一連の動作を実行する．まず，ヘッドが位置するます目の記号を読み取る．そして，この記号と現在の有限制御部の状態（内部状態）に依存して，このます目の記号を書き換え，ヘッドを左か右に 1 コマ移動し，内部状態を遷移させる．このような動作の規則を適切に定めることによって，いろいろな計算をチューリング機械に実行させることができる．そしてチューリングは，実行可能な「あらゆる計算」がこのチューリング機械という枠組みのなかで定式化できるのだというテーゼを主張した（チャーチ-チューリングの定立〔本章 4-6 参照〕）．彼の主張は「計算可能性」の概念の定義であるとも解釈できるが，この定義の適切性は今日に至るまで広く認められている．

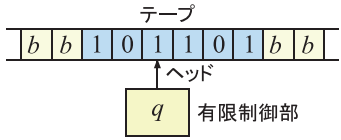

図 4･2　チューリング機械の概念図

チューリング機械は，形式的には $T = (Q, S, q_0, q_f, b, \delta)$ によって定義される．ここで，Q

は内部状態の有限集合，S はテープ記号の有限集合，$q_0\,(\in Q)$ は初期状態（動作を開始するときの状態），$q_f\,(\in Q)$ は最終状態（受理状態ともいい，出力または答が得られたことを表明する状態），$b\,(\in S)$ は空白記号でテープの有限個のます目を除く残りすべてのます目に b が書かれていると仮定する．δ は $Q\times S\times S\times\{L,R\}\times Q$ の部分集合であり，動作規則の集合を表す．δ の各要素は，$[q,s,s',d,q']$ のかたちの 5 項組で，これは T が状態 q で記号 s を読んだ場合，記号を s' に書き換え，ヘッドを d 方向に移動し（$d=L$ のときは左，$d=R$ のときは右），状態を q' に遷移させるという動作を表しており，機械に対するある種の命令と考えられる．もし T が状態 q で記号 s を読んでいるにもかかわらず，$[q,s,s',d,q']$ というかたちの 5 項組が δ 中に存在しないならば，T は停止する．

ここでもし，δ 中の任意の異なる二つの 5 項組 $[q_1,s_1,s_1',d_1,q_1']$ と $[q_2,s_2,s_2',d_2,q_2']$ に対し，$q_1=q_2$ ならば $s_1\neq s_2$ という関係が成り立つなら，T を決定性チューリング機械と呼ぶ．そうでないものを非決定性チューリング機械と呼ぶ．非決定性チューリング機械は，次に行う動作が 2 通り以上あり得るような機械である．

ここで次のような簡単なチューリング機械の例 $T_1=(Q_1,\{0,1\},q_0,q_f,0,\delta_1)$ を考える．ただし，$Q_1=\{q_0,q_1,\cdots,q_7,q_f\}$ であり，δ_1 は以下の 5 項組集合である．

$$
\begin{aligned}
\delta_1=\{\ & [\,q_0,0,0,R,q_1\,], && [\,q_1,0,0,R,q_f\,], && [\,q_1,1,0,R,q_2\,],\\
& [\,q_2,0,0,R,q_3\,], && [\,q_2,1,1,R,q_2\,], && [\,q_3,0,1,R,q_4\,],\\
& [\,q_3,1,1,R,q_3\,], && [\,q_4,0,1,R,q_5\,], && [\,q_5,0,0,L,q_6\,],\\
& [\,q_6,0,0,L,q_7\,], && [\,q_6,1,1,L,q_6\,], &&\\
& [\,q_7,0,1,R,q_1\,], && [\,q_7,1,1,L,q_7\,] && \}
\end{aligned}
$$

T_1 は，非負整数 n の単進表現（記号 1 を n 個並べたもの）が書かれたテープが与えられ，ヘッドを単進表現の左隣のます目に置いて初期状態 q_0 から動作を開始したとき，$f(n)=2n$ の単進表現を書き出して最終状態 q_f で停止する決定性チューリング機械である（図 4･3 参照）．

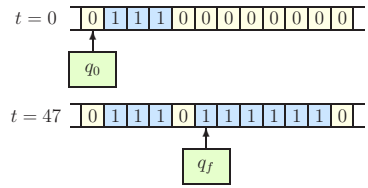

図 4･3　チューリング機械の例 T_1 による関数 $f(n)=2n$ の計算（$n=3$ の場合）．

この例から類推できるように，数の表現法を適切に定めることで，チューリング機械による関数の計算の概念を定式化できる．5 項組集合はいわばチューリング機械のためのプログラムであり，これを適切に与えることにより，実にいろいろな関数が計算できる．そして，チューリング機械で計算できる任意の関数は帰納的関数（recursive function）[5] と呼ばれる関数であり，逆に任意の帰納的関数はチューリング機械で計算できることが示されている．従って帰納的関数の族とチューリング機械で計算できる関数の族は一致する．また，チャーチ（A. Church）の λ 計算の体系[3]で計算できる関数族と一致することも知られている．これ

らにより，計算可能と考えられる関数はすべてチューリング機械で計算できるという，チューリングのテーゼの妥当性が更に裏付けられることになり，「計算万能性」を有するような理論的モデルのなかで最も標準的なものとなっている．

4-1-2 チューリング機械のバリエーション

チューリング機械には以上に述べた標準的なモデル以外にも多くのバリエーションが存在する．例えば，テープが左右両方向に無限なのではなく片方向にだけ無限であるようなモデル，テープを多数（従ってヘッドも多数）もつモデル，テープは 1 本だがヘッドを多数もつモデル，多次元のテープをもつモデル，テープ記号を 2 記号に制限したモデル，等々である．これらのモデルの各々は標準的なチューリング機械をシミュレートすることができ，その逆も成り立つことが知られているので，その意味ですべて等価である（ただし計算時間や使用メモリ量について論じる場合には差が生じることがある）．また，決定性と非決定性のチューリング機械も等価である．このため，これらのバリエーションは目的に応じて使い分けられる．

上記以外に，物理的な可逆性を反映したモデルとして，可逆チューリング機械がある．これは，計算過程を時間軸の逆方向に一意にたどれるようなモデルで，動作規則の集合 δ 中の任意の異なる二つの 5 項組 $[q_1, s_1, s_1', d_1, q_1']$ と $[q_2, s_2, s_2', d_2, q_2']$ に対し，$q_1' = q_2'$ ならば $d_1 = d_2$ かつ $s_1' \neq s_2'$ という関係が成り立つものをいう．先に述べたチューリング機械 T_1 は，可逆チューリング機械の例にもなっている．ベネット（C.H. Bennett）は，任意の非可逆なチューリング機械が，それと等価な可逆チューリング機械で計算終了時にテープにゴミ情報を残さないものに変換できることを示した[1)]．従って，可逆チューリング機械もまた計算万能である．

量子計算（量子オートマトン〔本編 2 章 2-5〕と量子計算〔本編 6 章 6-4〕参照）のモデルである量子チューリング機械[2, 4)]は，量子状態をとることができ，その時間発展（システム全体の状態遷移）はユニタリ演算子によって表現できる．ユニタリ演算子は全単射であるので，可逆チューリング機械は量子チューリング機械の特別な場合（量子状態ではなく古典的状態しかもたないような場合）といえる．量子チューリング機械などの量子計算のモデルは，量子状態を巧妙に用いることで，素因数分解など，いくつかの問題を高速に解くことができる．

■参考文献

1) C.H. Bennett, “Logical reversibility of computation,” IBM J. Res. Dev., vol.17, pp.525-532, 1973.
2) E. Bernstein and U. Vazirani, “Quantum complexity theory,” SIAM J. Computing, vol.26, pp.1411-1473, 1997.
3) A. Church, “An unsolvable problem of elementary number theory,” Am. J. Mathematics, vol.58, pp.345-363, 1936.
4) D. Deutsch, “Quantum theory, the Church-Turing principle and the universal quantum computer,” Proc. Roy. Soc. London Ser.A, vol.400, pp.97-117, 1985.
5) S.C. Kleene, “Introduction to Metamathematics,” North-Holland, Amsterdam, 1952.
6) E.L. Post, “Finite combinatory processes — Formulation 1,” J. Symbolic Logic, vol.1, pp.103-105, 1936.
7) A.M. Turing, “On computable numbers, with an application to the Entscheidungsproblem,” Proc. London Mathematical Society, Ser.2, vol.42, pp.230-265, 1936.

■6 群 - 2 編 - 4 章

4-2 句構造文法

（執筆者：森田憲一）[2009 年 1 月 受領]

句構造文法（phrase structure grammar）は，チョムスキー（N. Chomsky）が考案したある種の文法の体系であり，これによっていろいろな性質をもった記号列集合（形式言語）を生成することができる．チョムスキーはもともと，人の言語能力，特に統語的能力を記述するためにこれを考え出したのだが，その後計算機科学にとっても不可欠な枠組みとなった．本節では，句構造文法とその定義，及びチョムスキー階層を形成するそのサブクラスについて解説した後，句構造文法とオートマトンの関係を述べる．

4-2-1 句構造文法とチョムスキー階層

句構造文法は，チョムスキーの 1956 年の論文[1]のなかで提案されたもので，形式文法（formal grammar），生成文法（generative grammar）などとも呼ばれる．これは，有限個の記号列書き換え規則を用いて，記号列の集合（一般には無限集合になる）を生成するシステムである．

まず簡単な例を示す．(1) $S \to aSa$, (2) $S \to bSb$, (3) $S \to aa$, (4) $S \to bb$ の四つを書き換え規則としてもつ文法 G_0 を考える．ただし，S, a, b は基本記号で，特に S は非終端記号，a と b は終端記号と呼ばれる．このとき，記号 S から開始し，書き換え規則を (2),(2),(1),(4) の順に適用すると $S \overset{(2)}{\Rightarrow} bSb \overset{(2)}{\Rightarrow} bbSbb \overset{(1)}{\Rightarrow} bbaSabb \overset{(4)}{\Rightarrow} bbabbabb$ のように順に記号列が導出され，最後に $bbabbabb$ が得られる．書き換え規則の適用の仕方をいろいろと変えることによって，a と b からなる長さが偶数の回文（前から読んでも後ろから読んでも同じである記号列）がすべて生成できることが分かる．

Σ を記号の有限集合（アルファベット）とするとき，Σ に属する記号からなる有限列すべての集合（空列（長さ 0 の記号列）ε も含む）を Σ^* で表す．また，$\Sigma^+ = \Sigma^* - \{\varepsilon\}$ とする．句構造文法は，形式的には $G = (N, T, P, S)$ によって定義される．ここで，N は非終端記号の有限集合，T は終端記号の有限集合，P は書き換え規則の有限集合，S ($\in N$) は開始記号である．ここで P 中の各書き換え規則は $\varphi \to \psi$ （ただし $\varphi \in (N \cup T)^+$, $\psi \in (N \cup T)^*$ であり，φ は非終端記号を一つ以上含む）というかたちである．記号 S から開始し，書き換え規則を有限回適用して得られる終端記号列を，G によって生成される文と呼ぶ．また，G によって生成されるすべての文の集合を，G によって生成される言語と言い，$L(G)$ で表す．

上記のように定義される文法は，0 型文法（type-0 grammar）とも呼ばれる．チョムスキーは更に，書き換え規則のかたちに制限を加えることによって，次の三つのサブクラスを定義した[2]．

（1）1 型文法（文脈依存文法）

各書き換え規則が次のかたちである: $\varphi_1 A \varphi_2 \to \varphi_1 \omega \varphi_2$
（ただし $\varphi_1, \varphi_2 \in (N \cup T)^*$, $A \in N$, $\omega \in (N \cup T)^+$）

（2）2 型文法（文脈自由文法）

各書き換え規則が次のかたちである: $A \to \omega$ （ただし $A \in N$, $\omega \in (N \cup T)^+$）

（3）3 型文法（正規文法）

各書き換え規則が次のいずれかのかたちである: $A \to aB$ または $A \to a$

（ただし $A, B \in N$, $a \in T$）

上の定義より，$n = 0, 1, 2$ の各々について，n 型文法のクラスが $n+1$ 型文法のクラスを包含していることが容易に分かる．このような文法の包含関係をチョムスキー階層という．

4-2-2 句構造文法とチューリング機械

句構造文法が形式言語（すなわち記号列集合）の生成システムであるのに対し，チューリング機械などのオートマトンは，形式言語の受理装置（または認識装置）として定式化できる．チューリング機械の場合，入力記号列が書かれたテープを与えて初期状態から動作を開始させたとき，最終状態で停止するならば，この記号列が受理されたという（なお，入力記号列を受理しないのは，最終状態以外で停止する場合と，停止しないで永遠に動き続ける場合の 2 通りがある）．このようにして受理される記号列すべての集合が，そのチューリング機械 M によって受理される言語 $L(M)$ である．このとき，チューリング機械で受理される言語は 0 型文法で生成できる言語であり，その逆も成り立つことが証明できる．

この証明は，細部は多少複雑だが大きな困難はない．概略は以下のとおりである．句構造文法 G が与えられたとき，$L(G)$ を受理するチューリング機械 M は次のようにして作れる．M は G による記号列導出の過程を，開始記号からの導出のステップの短い順に，もれなく枚挙し，テープ上に書き出す．一つの導出過程を書き出すごとに，M への入力記号列と同じものが導出されているかを調べ，そうであれば入力を受理して停止する．そうでなければ枚挙を継続し，この操作を反復する．これにより $L(M) = L(G)$ となる M が得られる．

一方，チューリング機械 M に対し，$L(G) = L(M)$ となる句構造文法 G は次のようにして作れる．G の非終端記号として，M のテープ記号の各々と M の内部状態に対応する記号を用意する．G はまずそれらだけを使って，M の入力アルファベット上の任意の記号列を生成し（ただしそのコピーを残しておく），更に M の初期状態を表す非終端記号をヘッドの初期位置に作り出す．その後，G は M の動作を 1 ステップずつシミュレートするが，これは内部状態を表す記号とその前後の記号を書き換えるだけで行えるので，書き換え規則によって容易にシミュレートできる．M が最終状態に入ったならば，最初にとっておいたコピーを終端記号列に変え，それ以外の記号をすべて消去し，導出を終了する．これにより所望の G が得られる．

0 型文法によって生成される言語のクラスはまた，帰納的可算集合[4]のクラスとも一致する．ある集合 A が帰納的可算であるとは，A の要素をすべてもれなくテープ上に枚挙するようなチューリング機械 M が存在することをいう（A が無限集合なら M は停止せずに列挙を続ける）．実際，A を枚挙するチューリング機械を基に，A を受理するチューリング機械が構成でき，またその逆もいえることから，このような特徴づけが結論できる．

また，1–3 型の各文法クラスについても，それが生成する言語のクラスとちょうど一致するような受理言語のクラスをもつオートマトンのクラスが存在する．すなわち，1 型文法の言語生成能力は非決定性線形有界オートマトン（linear-bounded automaton）[3]によって，2 型文法は非決定性プッシュダウンオートマトンによって，3 型文法は有限オートマトンによって，それぞれ正確に特徴づけられる（表 4･1 参照）．

表 4·1　チョムスキー階層を形成する文法のクラスと，対応するオートマトン

文法のタイプ	名　称	対応するオートマトン
0 型	句構造文法	チューリング機械
1 型	文脈依存文法	非決定性線形有界オートマトン
2 型	文脈自由文法	非決定性プッシュダウンオートマトン
3 型	正規文法	有限オートマトン

■参考文献

1) N. Chomsky, “Three models for the description of languages,” IRE Trans. on Information Theory, vol.IT-2, pp.113-124, 1956.
2) N. Chomsky, “On certain formal property of grammars,” Information and Control, vol.2, pp.137-167, 1959.
3) S.Y. Kuroda, “Classes of languages and linear-bounded automata,” Information and Control, vol.7, pp.207-223, 1964.
4) H. Rogers, “Theory of Recursive Functions and Effective Computability,” McGraw-Hill, New York, 1967.

■6 群 - 2 編 - 4 章

4-3 万能チューリング機械

（執筆者：森田憲一）[2009 年 1 月 受領]

チューリング機械（Turing machine: TM）のなかには「あらゆる」チューリング機械をそれ 1 台でシミュレートできるような，そういう TM がある．これを万能チューリング機械（universal Turing machine: UTM）と呼ぶ．このような機械が構成できることはチューリング自身によって示されている[10]．本節では，UTM がどのように構成できるのか，その概略を述べた後，小サイズの UTM に関する話題を解説する．

4-3-1 万能チューリング機械の構成法の概略

UTM U は，任意の TM M の動作を模倣（シミュレート）できるような機械である．そのためには，M の記述と，M に与える入力の記述を，U に入力として与える必要がある．U はそれらを参照しながら M の動作を模倣し，M が最終的に出す答（出力）を U の答とするのである．M の記述が 5 項組集合（チューリング機械〔本章 4-1 参照〕）として与えられたならば，人が紙と鉛筆を使ってその動作を追うのは，いくぶん退屈で手間はかかるが，難しいことではない．UTM U はそのようなことを自身のテープ上で行うだけなので，基本的に困難な点はない．

U が模倣する TM M は，片無限テープをもち，記号数が 2 である決定性 TM であると仮定する（一般の TM はこのような TM で模倣できるので，このように制限しても差し支えない）．U のテープ上に，M の記述つまり 5 項組集合を蓄えるセクションと，M のテープを模倣する片無限のセクションを用意する（図 4･4 参照）．M の各 5 項組は，U が用いるテープ記号の列によって表現する．また各セクション中，M のヘッドの位置と，次に実行すべき M の 5 項組の位置に印をつけておく（図 4･4 では，それぞれ赤の三角と赤の丸印で表示）．U はまず，次に実行すべき 5 項組を読み取り，テープに書くべき記号とヘッドの移動方向を有限制御部に記憶したうえで，U のヘッドを M のヘッド位置まで移動する．そして，記号の書換えと M のヘッドの移動を模倣し，M の新しいヘッド位置にある記号を読んで有限制御部に蓄える．次に，今実行している 5 項組の位置まで U のヘッドを戻す．そうすると，次の時刻の M の状態が分かり，それと M が読んでいる記号とから，次に実行すべき 5 項組が決まるので，その位置に赤の丸印を移動する．以上で M の 1 ステップの動作の模倣が完了する．U はこの動作を M が停止するまで反復すればよい．U による M の直接的な模倣法は基本的には上のとおりであるが，多くのバリエーションがある．また後述のように，TM 以外の計算万能なシステムを模倣することによっても UTM が得られる．

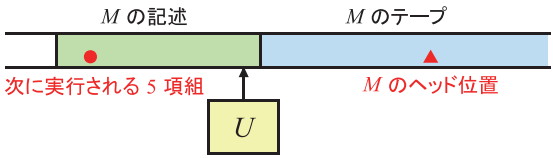

図 4･4　UTM U による TM M の模倣．

4-3-2 小サイズの万能チューリング機械

UTM でどれほど単純なものが存在するかは，理論的に大変興味深い問題である．シャノン（C.E. Shannon）は，テープ記号を 2 種類しか使わない（が多くの状態をもつ）UTM と，二つの状態しかもたない（が多数のテープ記号を使う）UTM が構成できることを示した[9]．また，UTM の複雑さの尺度として，状態数と記号数の積を用いることを提案した．これがきっかけとなり，以後，多くの研究者が小サイズの UTM を発見する問題に取り組んできた（これについてのサーベイは文献 15) に見られる）．

(1) 標準的な万能チューリング機械

標準的な TM は，無限長のテープをもつが，有限個のます目を除くすべてのます目に空白記号が書かれていると仮定している（有限性の条件）．この条件を満たすような小サイズの UTM にどんなものがあるかを見てみよう．

この研究の初期においては，TM を直接的に模倣する方法が多くとられた．渡辺による 8 状態 5 記号の UTM はそのなかでも小サイズのものである[11]．一方，サイズを更に小さくするため，間接的な方法も使われるようになった．つまり，あらゆる TM を模倣でき，かつそれよりも単純なシステムを模倣するのである．例えば，ある種の記号列書換えシステムである，2 タグシステム[3, 8]やサイクリックタグシステム[1]は計算万能であるので，これらを模倣できる TM を構成できれば UTM が得られる．

ミンスキー（M.L. Minsky）は，任意の 2 タグシステムを模倣するという方法で 7 状態 4 記号の UTM を構成し，それまでの記録を大幅に更新した[3]．ロゴジン（Y. Rogozhin）は，やはり 2 タグシステムを模倣する方法で，2 状態 18 記号，3 状態 10 記号，4 状態 6 記号，5 状態 5 記号，7 状態 4 記号，10 状態 3 記号，24 状態 2 記号の UTM を[8]，また，クトレック（M. Kudlek）とロゴジンは，3 状態 9 記号の UTM を得ている[2]．後に，ニアリ（T. Neary）とウッズ（D. Woods）は，タグシステムのバリエーションであるバイタグ（bi-tag）システムを用い，5 状態 5 記号，6 状態 4 記号，9 状態 3 記号，18 状態 2 記号の UTM を設計した[6]．図 4･5 の丸印は，少ない状態数または記号数の標準的 UTM の現時点での世界記録である．

また可逆 TM は，物理的な可逆性を反映した計算モデルである（チューリング機械〔本章 4-1 参照〕）が，これにも万能なものつまり，可逆 UTM が存在する．森田と山口は，任意のサイクリックタグシステムを模倣できるような 17 状態 5 記号[4]，及び 15 状態 6 記号[5]の可逆 UTM を与えた．

(2) 無限長の記号列を入力とする万能チューリング機械

これまでに述べた UTM はすべて有限性の条件を満たすものだった．しかしこの条件を緩め，無限個のます目に空白でない記号が書かれたテープを与えることを許すと，より単純な UTM が得られる．もちろん，入力として与える無限長記号列に何の制約も付加しないならば，普通の TM には計算できない関数も計算できてしまうことになり，具合が悪い（計算不可能な関数を一つとり，その値を表す無限の表をある種のオラクルとしてテープ上に与えておけばよい）．有限性の条件を緩和する良い方法の一つに，有限個のます目を除き記号列が周期的（ultimately periodic）と仮定するものがある．左方（または右方）にだけ周期的であり，右方（左方）には空白記号が続くような無限記号列を入力とする UTM を semi-weak UTM, 左右両方向に周期的（左と右とで周期パターンが異なっていてもよい）である無限記号列を入力とする UTM を weak UTM と呼ぶ．

渡辺は，任意の TM を直接模倣する 5 状態 4 記号と，7 状態 3 記号の semi-weak UTM を設計した[12]．また，ウッズとニアリは，サイクリックタグシステムを模倣する 3 状態 7 記号と，4 状態 5 記号の semi-weak UTM ををを与えた[14]．図 4･5 ではこれらはひし形で示されている．

ウォルフラム（S. Wolfram）は，規則番号 110 の初等的セルオートマトンと呼ばれる，単純な遷移規則をもつ 1 次元 2 状態 3 近傍のセルオートマトンを模倣することにより，状態数と記号数の非常に少ない weak UTM が得られる可能性を指摘した[13]．クック（M. Cook）は，任意のサイクリックタグシステムが規則番号 110 の初等的セルオートマトンのなかで模倣できることを示し，これの計算万能性を証明した[1]．クックは更に，このセルオートマトンを模倣するという方法で，2 状態 5 記号，3 状態 4 記号，4 状態 3 記号，7 状態 2 記号の weak UTM を与えた[1]．ニアリとウッズは後にこれを改良し，2 状態 4 記号，3 状態 3 記号，6 状態 2 記号の weak UTM を設計した[7]．図 4･5 ではこれらは四角形で示されている．

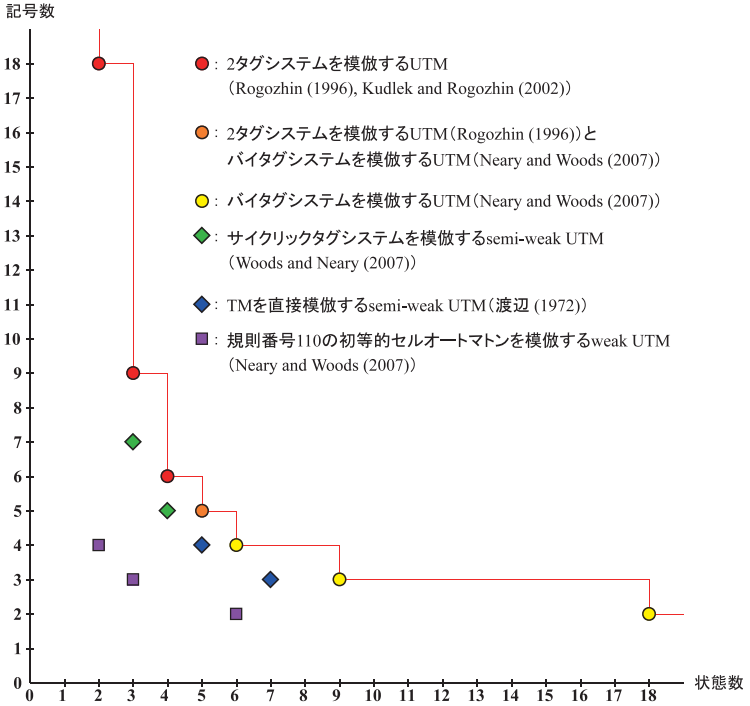

図 4･5　これまでに知られている小サイズの UTM, semi-weak UTM 及び weak UTM（文献 15) による）赤の実線を含みそれより右上は，有限性の条件を満たす（標準的な）UTM の存在が示されている領域．

またウォルフラムは，非常に単純な特定の 2 状態 3 記号 TM が UTM であると予想し，懸賞問題とした．この問題は，2007 年に英国の学生スミス（A. Smith）によって肯定的に解かれた（問題と解は `http://www.wolframscience.com/prizes/tm23/` 参照）．スミスの方

法では，この TM に与える入力は，有限個のます目を除いて周期的，という条件にも合致しない無限記号列（ただしある定まった方法で構成される）になるため，異論もある．

■参考文献

1) M. Cook, “Universality in elementary cellular automata,” Complex Systems, vol.15, pp.1-40, 2004.
2) M. Kudlek and Y. Rogozhin, “A universal Turing machine with 3 states and 9 symbols,” Proc. DLT 2001, LNCS-2295, Springer-Verlag, pp.311-318, 2002.
3) M.L. Minsky, “Computation: Finite and Infinite Machines,” Prentice-Hall, Englewood Cliffs, NJ, 1967.
4) K. Morita and Y. Yamaguchi, “A universal reversible Turing machine,” Proc. 5th Int. Conf. on Machines, Computations, and Universality, LNCS-4664, Springer-Verlag, pp.90-98, 2007.
5) K. Morita, “Reversible computing and cellular automata — a survey,” Theoret. Comput. Sci., vol.395, pp.101-131, 2008.
6) T. Neary and D. Woods, “Four small universal Turing machines,” Proc. 5th Int. Conf. on Machines, Computations, and Universality, LNCS-4664, Springer-Verlag, pp.242-254, 2007.
7) T. Neary and D. Woods, “Small weakly universal Turing machines,” arXiv:0707:4489v1[cs.CC], 2007.
8) Y. Rogozhin, “Small universal Turing machines,” Theoret. Comput. Sci., vol.168, pp.215-240, 1996.
9) C.E. Shannon, “A universal Turing machine with two internal states,” Automata Studies, Princeton University Press, Princeton, NJ, pp.157-165, 1956.
10) A.M. Turing, “On computable numbers, with an application to the Entscheidungsproblem,” Proc. London Mathematical Society, Ser.2, vol.42, pp.230-265, 1936.
11) 渡辺茂, “万能 TURING 機械の最小サイズについて,” 情報処理, vol.2, no.3, pp.131-137, 1961.
12) 渡辺茂, “4 記号 5 状態の万能 TURING 機械,” 情報処理, vol.13, No.9, pp.588-592, 1972.
13) S. Wolfram, “A New Kind of Science,” Wolfram Media Inc., 2001.
14) D. Woods and T. Neary, “Small semi-weakly universal Turing machines,” Proc. 5th Int. Conf. on Machines, Computations, and Universality, LNCS-4664, Springer-Verlag, pp.303-315, 2007.
15) D. Woods and T. Neary, “The complexity of small universal Turing machines: A survey,” Theoret. Comput. Sci., 2008, doi:10.1016/j.tcs.2008.09.051, in press.

■6 群 - 2 編 - 4 章

4-4 停止問題と決定不能性

（執筆者：小林孝次郎）［2008 年 12 月 受領］

あるアルゴリズムとあるデータが与えられたときに，そのアルゴリズムにそのデータを与えて動かしたときにそのアルゴリズムが停止するか否かを決定する問題を，（アルゴリズムの）**停止問題**（halting problem）と呼ぶ．この問題は，それを解くアルゴリズムが存在しない，という意味で，決定不能であることが知られている．決定不能問題にはいろいろなものがあるが，停止問題が決定不能であるというこの結果は，これらのたくさんの問題が決定不能であることを証明する出発点となる．停止問題とそれが決定不能であるという結果をもう少し厳密に述べると，以下のようになる．

「アルゴリズム」という直観的な概念を厳密な概念として定義する方法にはいろいろなものがあるが〔本章 4-6 参照〕，ここではチューリング機械を「アルゴリズム」の定義として用いることにしよう〔本章 4-1 参照〕．またチューリング機械は，自然数を変数の値及び関数の値としてとるような関数を計算するために用いることにする．そうすると，「アルゴリズムによって計算することができる（自然数に関する）関数」という直観的な概念は，「チューリング機械によって計算できる関数」と定義される．このような関数を，**計算可能関数**（computable function）と呼ぶ．また「アルゴリズムによって決定できる（自然数に関する）述語」という直観的な概念は，「真」を 1，「偽」を 0 で表したとき，「チューリング機械によってその真偽を計算できる関数」と定義される．このような述語を，**決定可能述語**（decidable predicate）と呼ぶ〔本章 4-1 参照〕．

すべてのチューリング機械に，自然数による番号を割りつけることができる．そのためには，チューリング機械の構造をあるアルファベットの文字列で表現する方法を決めておき，それらの表現を辞書式順序（長さの短い文字列は長い文字列より前に，また同じ長さの文字列は辞書のなかで前にくるものが前にくる，と定義した文字列の間の順序）で並べ，それらに 0, 1, 2, … と番号をつけていけばよい．

この番号を用いると，上に述べた「停止問題は決定不能である」という結果は，次のように定義される関数 $f_{\mathrm{halt}}(i, x)$ が計算不能関数（計算可能でない関数）である，と述べることができる:

$$f_{\mathrm{halt}}(i, x) = \begin{cases} 1 & \text{番号 } i \text{ のチューリング機械に } x \text{ をデータとして与えると停止するとき,} \\ 0 & \text{停止しないとき.} \end{cases}$$

この結果は，以下のような順序で証明することができる．

(Step 1) $f_{\mathrm{univ}}(i, x)$ という部分関数（「関数」の概念を，場合によっては値が定義されないことを許すように拡張したもの）を次のように定義する:

$$f_{\mathrm{univ}}(i, x) \quad = \quad \text{番号 } i \text{ のチューリング機械に } x \text{ をデータとして与えたときの出力}$$
$$\text{（ただし計算が停止しない場合には値は定義されない）．}$$

この部分関数は，計算可能部分関数であることが知られている（部分関数が計算可能である

とは，値が定義されているときにはその値を出力して停止し，値が定義されていないときには停止しない，という意味でそれを計算するチューリング機械が存在することと定義する）．これは，万能チューリング機械と呼ばれるチューリング機械が存在することから分かる（本章 4-3 節参照）．

（Step 2） $f_{\mathrm{diag}}(i)$ という関数を次のように定義する:

$$f_{\mathrm{diag}}(i) = \begin{cases} f_{\mathrm{univ}}(i,i)+1 & f_{\mathrm{univ}}(i,i) \text{ が定義されるとき,} \\ 0 & \text{定義されないとき.} \end{cases}$$

この $f_{\mathrm{diag}}(i)$ は計算可能関数ではない．このことは，次のように背理法で示せる．

$f_{\mathrm{diag}}(i)$ が計算可能関数であると仮定しよう．そうすると $f_{\mathrm{diag}}(i)$ を計算するチューリング機械が存在する．その番号を i_0 とすると，すべての i に対し $f_{\mathrm{diag}}(i) = f_{\mathrm{univ}}(i_0, i)$ が成り立つ．ここで i に i_0 を代入すると，$f_{\mathrm{diag}}(i_0) = f_{\mathrm{univ}}(i_0, i_0)$ という式が得られる．しかし，この式から以下のように矛盾が生じる．もし $f_{\mathrm{univ}}(i_0, i_0)$ が定義されるなら，その値を s とすると，この式の左辺は $s+1$ であるが右辺は s である．一方もし $f_{\mathrm{univ}}(i_0, i_0)$ が定義されないなら，左辺は 0 であるが右辺は定義されない．このように，いずれの場合にも矛盾が生ずる．従って $f_{\mathrm{diag}}(i)$ を計算するチューリング機械は存在せず，$f_{\mathrm{diag}}(i)$ は計算可能関数ではない．ここで用いた証明法を**対角線論法**（diagonalization）と呼ぶ．

（Step 3） $f_{\mathrm{halt}}(i, x)$ が計算可能関数でないことを，背理法で示す．関数 $f_{\mathrm{halt}}(i, x)$ が計算可能関数であると仮定しよう．$f_{\mathrm{diag}}(i)$ の定義は次のように書き換えることができる:

$$f_{\mathrm{diag}}(i) = \begin{cases} f_{\mathrm{univ}}(i,i)+1 & f_{\mathrm{halt}}(i,i) = 1 \text{ のとき,} \\ 0 & f_{\mathrm{halt}}(i,i) = 0 \text{ のとき.} \end{cases}$$

従って，任意の与えられた i に対して，$f_{\mathrm{diag}}(i)$ の値を次の方法で計算できる．まず f_{halt} を計算するチューリング機械を用いて $f_{\mathrm{halt}}(i, i)$ の値を計算する．その値が 1 であれば，f_{univ} を計算するチューリング機械を用いて $f_{\mathrm{univ}}(i, i)$ の値を計算し，その値に 1 を加えた値を出力する．$f_{\mathrm{halt}}(i, i)$ の値が 0 であれば，0 を出力する．この計算があるチューリング機械によって実行できることは明らかであるから，$f_{\mathrm{diag}}(i)$ は計算可能関数である．これは（Step 2）の結果に矛盾する．

ある関数 f が計算不能であることが分かっているとき，もし関数 g が計算可能なら f も計算可能になるということを示すことによって，g が計算不能であることを証明することができる．この証明法を，f の g への**還元**（reduction）と呼ぶ．上記の証明では，まず $f_{\mathrm{diag}}(i)$ という計算不能関数が対角線論法で作られた後，f_{diag} の f_{halt} への還元により f_{halt} が計算不能であることが示されている．このように，$f_{\mathrm{diag}}(i)$ から出発して次々と還元を行っていくことによって，色々な関数，述語が計算不能，決定不能であることを示すことができる．

■参考文献

1) 鹿島 亮, “C 言語による計算の理論,” サイエンス社, 2008.
2) 小林 孝次郎, “計算論,” コロナ社, 2008.

■6 群 - 2 編 - 4 章

4-5 帰納的可算

（執筆者：小林孝次郎）［2008 年 12 月 受領］

自然数の集合 X は，X のすべての要素を次々と重複なく出力してゆくアルゴリズムが存在するとき，つまり $X = \{x_0, x_1, x_2, \ldots\}$ であるような異なる数 $x_0, x_1, x_2, \ldots$ を次々と出力してゆくアルゴリズムが存在するとき，**帰納的可算集合**（recursively enumerable set）であるという．ただしこの定義では，有限集合は帰納的可算集合になり得ないので，有限集合はすべて帰納的可算集合である，と定義を修正する．この概念は，自然数の集合だけではなく，文字列やグラフなどの集合に対しても同じように定義できる．

「アルゴリズム」という直観的な概念は，例えばチューリング機械を用いて厳密な概念として定義することができる〔本章 4-1 参照〕．自然数を変数の値及び関数の値としてとる部分関数 $f(x)$ は，x を入力として与えたとき，$f(x)$ が定義される場合には $f(x)$ の値を出力して停止し，定義されない場合にはいつまでも停止しない，というチューリング機械が存在するとき，**計算可能部分関数**（computable partial function）であるという（部分関数とは，「関数」の概念を，場合によっては値が定義されないことを許すように拡張したものである）．

計算可能部分関数の概念を用いると，自然数の集合 X は，有限集合であるか，あるいは (i) $x \neq x'$ なら $f(x) \neq f(x')$, (ii) $X = \{f(0), f(1), f(2), \ldots\}$, という二つの条件を満足する計算可能関数（計算可能部分関数であって関数でもあるもの）$f(x)$ が存在するとき帰納的可算集合である，と厳密に定義することができる．「帰納的」という言葉を用いるのは，計算可能関数は歴史的に，「帰納的関数」（recursive function）とも呼ばれるからである〔本章 4-6 参照〕．

上記の帰納的可算集合の定義は，集合の要素を「数え上げる」という意味で，「帰納的可算」という言葉の直観的意味とよくあっているが，帰納的可算集合は，これ以外にもいろいろな方法で特徴づけることができる．以下にそのいくつかを示す．

- 集合 X は，空集合であるか，$X = \{f(0), f(1), f(2), \ldots\}$ が成り立つような計算可能関数 $f(x)$ が存在するとき，そのときに限り帰納的可算である
- 集合 X は，計算可能部分関数 $f(x)$ でその値域（$f(x)$ の値が定義されるような x に対する $f(x)$ の集合）が X と一致するようなものが存在するとき，そのときに限り帰納的可算である
- 集合 X は，計算可能部分関数 $f(x)$ でその定義域（$f(x)$ の値が定義されるような x の集合）が X と一致するようなものが存在するとき，そのときに限り帰納的可算である
- 集合 X は，$X = \{x \mid \exists y[f(x,y) = 0]\}$ が成り立つような計算可能関数 $f(x,y)$ が存在するとき，そのときに限り帰納的可算である

自然数の集合 X は，$x \in X$ なら $f(x) = 1$, $x \notin X$ なら $f(x) = 0$ であるような計算可能関数 $f(x)$ が存在するとき，**決定可能集合**（decidable set）であるという．次の定理は，決定可能集合と帰納的可算集合の関係を示す．記号 $\overline{X}$ は X の補集合を表す．

定理 1 集合 X は，X と $\overline{X}$ がともに帰納的可算であるとき，そのときに限り決定可能である．

この定理から，決定可能集合はすべて帰納的可算集合であることが分かる．一方，帰納的可算ではあるが決定可能ではない集合も存在する．すべてのチューリング機械には，本章 4-4 で示したように，自然数による番号がつけられているものとする．

定理 2 次のように定義される集合 X_{halt} は，帰納的可算ではあるが決定可能ではない：

$$X_{\mathrm{halt}} \quad = \quad \{(i, x) \mid \text{番号 } i \text{ のチューリング機械に } x \text{ を入力として与えると停止する}\}.$$

定理 3 集合 X_{total} を次のように定義すると，X_{total} も $\overline{X_{\mathrm{total}}}$ も帰納的可算集合ではない：

$$X_{\mathrm{total}} \quad = \quad \{i \mid \text{番号 } i \text{ のチューリング機械は，どのような入力を与えても停止する}\}.$$

集合 X の性質と $x \in X$ が成り立つか否かを調べるアルゴリズムの関係を次に示す．

(1) X が決定可能であるとき（例えば偶数の集合）：この場合は，x を与えたとき，$x \in X$ なら “yes” を，$x \notin X$ なら “no” を出力して停止するアルゴリズムが存在する．従って，計算時間を気にしなければ，$x \in X$ が成り立つか否かは，アルゴリズムによって完全に決定できる．

(2) X は帰納的可算ではあるが $\overline{X}$ は帰納的可算でないとき（例えば X_{halt}）：この場合，$x \in X$ なら “yes” を出力して停止し，$x \notin X$ ならいつまでも停止しない，というアルゴリズムは存在する．$x \in X$ なら必ず “yes” という出力が得られるが，計算がまだ続いている時点では，もう少し待てば “yes” という出力が得られるのか，それともいつまで待っても停止しないのかは，判断できない．(1) 型のアルゴリズムも，次に示す (3) 型のアルゴリズムも存在しない．

(3) X は帰納的可算でないが $\overline{X}$ は帰納的可算であるとき（例えば $\overline{X_{\mathrm{halt}}}$）：この場合，$x \in X$ ならいつまでも停止せず，$x \notin X$ なら “no” を出力して停止する，というアルゴリズムは存在する．(1) 型のアルゴリズムも (2) 型のアルゴリズムも存在しない．

(4) X も $\overline{X}$ も帰納的可算でないとき（例えば X_{total}）：この場合，(1) 型，(2) 型，(3) 型のいずれのアルゴリズムも存在しない．

従って，(1) 型 ⟶ (2) 型 ⟶ (4) 型, (1) 型 ⟶ (3) 型 ⟶ (4) 型と進むにつれ，$x \in X$ が成り立つか否かを決定する問題はだんだん難しくなる．

形式言語理論から例をとると，$\{(G_1, G_2) \mid L(G_1) \cap L(G_2) \neq \emptyset\}$, $\{(G_1, G_2) \mid L(G_1) \neq L(G_2)\}$, $\{G \mid L(G) \neq \Sigma^*\}$ は，帰納的可算であるが決定可能ではない集合の例である．ただしここで G などは文脈自由文法を表し，$L(G)$ などは G などが生成する言語を表し，Σ は，$L(G) \subseteq \Sigma^*$ であるようなアルファベットとする〔2 編 1 章 1-2, 2 編 3 章 3-1 参照〕．

■参考文献

1) 鹿島亮, “C 言語による計算の理論,” サイエンス社, 2008.
2) 小林孝次郎, “計算論,” コロナ社, 2008.

■6 群 - 2 編 - 4 章

4-6 チャーチ・チューリングの定立

（執筆者：小林孝次郎）［2008 年 12 月 受領］

問題を解くための計算手順という意味での「アルゴリズム」という言葉は，数学や情報科学の分野でよく使われる．例えば次に示すのは，与えられた自然数 n が素数か否かを判定するアルゴリズムである．まず $n = 0$ か $n = 1$ であれば “no” を出力し，$n = 2$ なら “yes” を出力して終了する．$n \geq 3$ であれば，$i = 2, 3, \ldots, n-1$ で n を割ってみる．そのなかに n を割り切る i が存在すれば “no” を，存在しなければ “yes” を出力して終了する．

この「与えられた自然数が素数か否かを判定する」という問題に対しては，（効率を気にしなければ）それを解くアルゴリズムが簡単に求まった．しかし問題によっては，アルゴリズムを発見するのにかなりの努力を必要とする．

1930 年代に，いくつかの問題に対し，それを解くアルゴリズムが存在しないのではないかという立場から研究が行われはじめた．このような問題の例として，**ヒルベルトの第 10 問題**がある．この問題は，整数係数の多変数の多項式 $f(x_1, \ldots, x_n)$ が与えられたとき，方程式 $f(x_1, \ldots, x_n) = 0$ が整数解 $x_1, \ldots, x_n$ をもつか否かを判定するアルゴリズムを示せ，という問題である．例えば，$3x_1x_2^3x_3^2 - 4x_2x_3x_4^2 + 11x_2^2 - 41x_1x_2x_3 + 18 = 0$ というような方程式が与えられ，この方程式を成り立たせる整数 $x_1, \ldots, x_4$ が存在するか否かを決定することが求められる．少し自分で考えてみると分かるが，この問題のアルゴリズムを見つけることはなかなか難しい．

ある問題を解くアルゴリズムが存在しないことを数学的に厳密に証明するためには，その前にまず，「アルゴリズム」という直観的な概念を数学的に定義する必要がある．これに対しいくつかの数学的な定義が提案されたが，結果としてそれらはすべて同値であることが判明した．**チャーチ・チューリングの定立**（Church-Turing’s Thesis）は，これらの数学的に同値な定義は「アルゴリズム」という直観的な概念の適切な定義になっていると考えてよい，という主張である．現在では，この主張は妥当であると考えられている．この主張は直観的な概念と数学的な概念の間の関係に関するものであるから，それが正しいことを数学的に証明することはできない．

「アルゴリズム」の数学的定義として提案されたものには，チューリング機械，ラムダ計算，帰納的関数などがある．**チューリング機械**は本章 4-1 で説明されている．**ラムダ計算**（lambda calculus）は，関数の組み合わせによってより複雑な関数を構成するためのある種の機構であり，これを用いてアルゴリズムの概念を定義することができる[1)]．以下では，**帰納的関数**（recursive function）について説明する．

帰納的関数とは，いくつかの基本関数から出発して，関数から関数を作り出すいくつかの基本演算を繰り返し適用することによって得られる関数のことである（関数としては，変数の値と関数の値がともに自然数であるものを考える）．

出発となる基本関数は，次の 3 種類である．

- $Z(x_1) = 0$,
- $S(x_1) = x_1 + 1$,
- $P_i^n(x_1, \ldots, x_n) = x_i$ $(n \geq 1, 1 \leq i \leq n)$.

関数から関数を作り出す基本演算は，次の三つの演算である．

(i) 代入: これは，関数 $f(y_1,\ldots,y_m)$，$g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n)$ から

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

という関数を作り出す演算である．

(ii) 原始帰納法: これは，関数 $f(x_2,\ldots,x_n)$, $g(x_1,\ldots,x_n,y)$ から

$$h(0,x_2,\ldots,x_n) = f(x_2,\ldots,x_n),$$

$$h(x_1+1,x_2,\ldots,x_n) = g(x_1,x_2,\ldots,x_n,h(x_1,x_2,\ldots,x_n))$$

で定義される関数 $h(x_1,x_2,\ldots,x_n)$ を作り出す演算である．

(iii) 最小化: これは，どのような $x_1,\ldots,x_n$ に対しても $f(x_1,\ldots,x_n,y)=0$ が成り立つような y が存在するような関数 $f(x_1,\ldots,x_n,y)$ から

$$g(x_1,\ldots,x_n) = (f(x_1,\ldots,x_n,y)=0 \text{ が成り立つ最小の } y)$$

という関数を作り出す演算である．

基本関数から出発して基本演算を繰り返し適用してある関数を作り出す過程を記述したものは，その関数を計算するアルゴリズムとみなすことができる．アルゴリズムをこのような記述と定義すれば，アルゴリズムによって計算できる関数とは帰納的関数のことである．

アルゴリズムの定義として帰納的関数を用いるとき，チャーチ・チューリングの定立は

関数 $f(x_1,\ldots,x_n)$ はアルゴリズムによって計算できる

$\Longleftrightarrow$ 関数 $f(x_1,\ldots,x_n)$ は帰納的関数である

ということが成り立つと考えてよい，という主張である．ここで $\Longleftarrow$ 方向が成り立つことについては，ほぼ異論はないであろう．$f(x_1,\ldots,x_n)$ を帰納的関数として作る過程の記述は，確かにある種のアルゴリズムと考えることができる（プログラミングの言葉でいえば，原始帰納法は `for` ループで自然に実現でき，最小化は `while` ループで自然に実現できる）．そこでチャーチ・チューリングの定立は，実質的には $\Longrightarrow$ 方向が成り立つと考えてよい，という主張である．もし $\Longrightarrow$ 方向が成り立たないとすると，たとえある関数が帰納的関数でないことが証明できたとしても，それによってその関数を計算するアルゴリズムが存在しないことが証明できた，と信じるわけにいかない．

チューリング機械などがアルゴリズムの定義として提案されたのは，まだコンピュータが存在しない時代のことであった．もし現在このような定義を考えるならば，我々は迷わず「アルゴリズムとはコンピュータのプログラムのことである」と定義するであろう．その場合でも，プログラムを表現するのにどのプログラミング言語を用いるかが問題になるが，

- 利用できるデータは整数型だけである
- 整数の桁数，配列のサイズ，再帰呼び出しの深さ，などには制限を設けない

というような修正を施せば，どのプログラミング言語を用いても同値な定義が得られる．そして得られた定義は，チューリング機械などを用いる従来の定義とも同値である（C 言語を用いたこのような試みについては文献 1) を参照）．現在では我々は，計算とはコンピュータで実行するもの，という考えに慣れてしまっているので，アルゴリズムを「コンピュータのプログラム」と定義することに違和感はなく，この定義に対するチャーチ・チューリングの定立のようなものの必要性もほとんど感じない．

■参考文献

1) 鹿島 亮, “C 言語による計算の理論,” サイエンス社, 2008.