1　帰納的関数とチューリング機械と決定不能問題

1.1　帰納的関数

これから、定義域が自然数 (もしくは非負整数) 全体又はその部分集合で、値域が自然数 (もしくは非負整数) であるような関数を考える。自然数全体で値が定義されている関数を全域関数と呼ぶ。一般にそうとは限らない場合、部分関数と呼ぶ。

きわめて初等的な関数の組を出発点にして、関数から関数を構成、定義する操作をいくつか導入し、それを有限回繰り返して得られる関数の全体のなすクラスを定義する。このような形式で定義される関数のクラスとして、原始帰納的関数、帰納的関数[1] のクラスを以下で定義する。帰納的関数は、「人間が計算することのできる関数」のクラスと見なせるものである。

原始帰納的関数

Def. 1 原始帰納的関数。次の初等的関数のリストから始めて、合成と原始帰納法を有限回繰り返して得られる関数を原始帰納的関数という。このクラスは、普通に現れる算術関数をほとんどすべて含んでいる。

(1) $S(x) = x + 1$,

(2) $N(x) = 0$,

(3) $U_i^n(x_1; \cdots ; x_n) = x_i \quad (1 \leq i \leq n)$.

ここで、既知の関数の組から新しい関数を定義する演算である合成と原始帰納法は、次で定義される。

(1) 関数の合成。自然数もしくはその部分集合上の関数 $f(x_1; \cdots ; x_m); g_1(x_1; \cdots ; x_n); \cdots ; g_m(x_1; \cdots ; x_n)$ があるとき

$$h(x_1; \cdots ; x_n) = f(g_1(x_1; \cdots ; x_n); \cdots ; g_m(x_1; \cdots ; x_n))$$

で新しい関数 h を作る。ただし、 h の値は右辺が定義できるときに限り定義されているものとする。

(2) 原始帰納法。全域関数 $f(x_1; \cdots ; x_n); g(y_1; \cdots ; y_{n+2})$ に対して次のスキーマ (数学的帰納法) で関数 h を定義する。

$$\begin{aligned} h(0; x_1; \cdots ; x_n) &= f(x_1; \cdots ; x_n) \\ h(z+1; x_1; \cdots ; x_n) &= g(z; h(z; x_1; \cdots ; x_n); x_1; \cdots ; x_n) \end{aligned}$$

言い換えると、恒等写像 U_i^n、successor 関数 S、定数関数 N があって、関数の合成と数学的帰納法が使えれば、ふつうに現れる算術関数はほとんどすべて構成できると分かる。

例 1 具体的な原始帰納的関数の構成の例。

[1] 最初 Gödel が recursive function の名で計算できる初等的な関数のクラスをひとつ定義した。これはここで言う原始帰納的関数である。ついで Kleene がより一般的な関数のクラスを研究し、Gödel の recursive function と区別するため、general recursive function と呼んだ。これがここで言う帰納的関数である。

(1) $A(x;y) = x + y$。 次の原始帰納法から定義できる。

$$\begin{cases} A(x;0) = U_1^1(x) & (= x) \\ A(x;y+1) = S(A(x;y)) & (= (x+y)+1) \end{cases}$$

(2) $M(x;y) = xy$。 次の原始帰納法で定義できる。

$$\begin{cases} M(x;0) = N(x) & (= 0) \\ M(x;y+1) = A(M(x;y);U_1^2(x;y)) & (= M(x;y) + x = xy + x) \end{cases}$$

(3) 単位の減算の関数

$$P(x) = \begin{cases} x - 1 & x > 0 \text{ のとき} \\ 0 & x = 0 \text{ のとき} \end{cases}$$

は、次の原始帰納法で定義される。

$$\begin{cases} P(0) = N(x) \\ P(x+1) = U_1^1(x) \end{cases}$$

(4) 非負整数の引き算

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

は、次の原始帰納法から定義される。

$$\begin{cases} x \dot{-} 0 = U_1^1(x) \\ x \dot{-} (y+1) = P(x \dot{-} y) \end{cases}$$

演習 1 上の例を参考にして、以下の関数が原始帰納的であることを示せ。

(1) $n!$ は次の帰納法で定義できる。

$$\begin{cases} F(n) = 1 \\ F(n) = M(n; F(n-1)) \end{cases}$$

(2) $|x - y|$, $|x - y| = (x \dot{-} y) + (y \dot{-} x)$

(3) $f(x;y) = x^y$.

$$\begin{cases} f(x;0) = 1; \\ f(x;y+1) = M(f(x;y); x) \quad (= f(x;y) \cdot x) \end{cases}$$

帰納的関数　ここで、つぎのような別の初等的な関数のリストを考える。

(1) $S(x) = x + 1$,

(2) $U_i^n(x_1; \cdots; x_n) = x_i \qquad (1 \leq i \leq n)$,

(3) $x + y$,

(4) $x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y; \\ 0 & \text{otherwise;} \end{cases}$

(5) xy

Def. 2 関数を構築するもう一つの演算として関数の最小化を次のように定義する。$f(y; x_1; \cdots; x_n)$ を全域関数とする。$x_1; \cdots; x_n$ が与えられたとき、$f(y; x_1; \cdots; x_n) = 0$ を満たす y が存在すれば、そのような y の値の最小値を値として持ち、それ以外では定義されない関数を $h(x_1; \cdots; x_n)$ とすると、これを f の最小化といい、

$$h(x_1; \cdots; x_n) = \min_y (f(y; x_1; \cdots; x_n) = 0)$$

と書くことにする。

Def. 3 帰納的部分関数 (partial recursive function)
上のリストの関数を出発点にして、合成と最小化の演算を有限回適用して得られる関数を帰納的部分関数という。

Def. 4 帰納的関数
帰納的部分関数を作る操作で、結果が全域で定義されるときに限って最小化の演算をほどこすとしたときに定義される全域関数を帰納的関数と呼ぶ。

補題 1 任意の整数 $a_0; a_1; \cdots; a_n$ に対して帰納的関数 T_i $(i = 0; 1; \cdots; n)$ とある数 w が存在して $T_i(w) = a_i$ $(i = 0; 1; \cdots; n)$ となる。

定理 1 原始帰納的関数は、帰納的関数である。

(Proof) $f(x_1; \cdots; x_n)$ と $g(y_1; \cdots; y_{n+2})$ が帰納的関数で、関数 $h(z; x_1; \cdots; x_n)$ がそれらから原始帰納法で定義されるとする。このとき h も帰納的であることを示す。補題 1 から

$$T_0(w) = f(x_1; \cdots; x_n) \quad (= a_0)$$
$$T_i(w) = g(i; T_{i-1}(w); x_1; \cdots; x_n) \quad (= a_i) \quad (i = 0; 1; \cdots; y)$$

となる数 w が存在する。ゆえに

$$h(y; x_1; \cdots; x_n) = T_y\left(\min_w \left(T_0(w) = f(x_1; \cdots; x_n)\right) \wedge \bigwedge_{z=0}^{y-1} \left(T_{z+1}(w) = g(z; T_z(w); x_1; \cdots; x_n)\right)\right)$$

と定義すると、$h(y; x_1; \cdots; x_n) = T_y(w) = g(y - 1; T_{y-1}(w); x) = \cdots$ となって帰納法による定義と一致することが分かる。 □

逆は成立しない。(2変数の帰納的関数 $f(n; x)$ で任意の1変数原始帰納的関数 $h(x)$ に対してある n が存在して、$h(x) = f(n; x)$ となるものがあることが証明できる。ここで f が原始帰納的であれば、$g(x) = f(x; x)+1$ も原始帰納的になり、矛盾である。)

1.2 チューリング機械

問題を解くための有効な手続きもしくはアルゴリズム (解法) という直感的な概念を数学的に定式化するもののひとつとしてチューリング機械がある。

自然数上の関数は、それを計算するためのアルゴリズム (有効な手続き) が存在する計算可能な関数とそうでない計算不能な関数に分けられるが、実は計算不能な関数は非常にたくさんあることが簡単に分かる。実際、自然数から f0; 1g の上への関数は実数と1対1対応するので、全体は連続無限の集合になる。一方、アルゴリズムはそれぞれ長さ有限の記号列で記述されるので、全体としても高々可算個である。ゆえに計算不能な関数が連続無限個以上ある。

チューリング機械は 1936 年 Alan Turing によって創案された。それは有限制御部と入力テープとテープヘッドを持つ。入力テープはます目 (cell) に分かれていて、テープヘッドは1回にひとつのマス目を読み書きする。テープは片方向無限で右側に無限に延びているとする。

テープは最初左端から有限個の記号がマス目にならび、その残りのマス目は空白と呼ばれる特別な記号 B で埋められている。チューリング機械は各時点で、内部状態とテープヘッドが読んでいるマス目の記号に従い次の動作を同時に行ない、次の時点へ移る。

(1) テープヘッドが見ているマス目の記号を書き換える。

(2) テープヘッドを右又は左隣のマス目に移動する。

(3) 内部状態を変える。

形式的にチューリング機械を定義すれば、それは $M = (Q; §; ¡; ±; q_0; B; F)$ と表される。ここで、

- Q は内部状態の集まりの有限集合。
- ¡ は可能なテープ記号の有限集合で、空白記号 B を含む。
- § は入力記号の集合で、B を含まない ¡ の部分集合。
- ± は動作関数。$± : Q £ ¡ ! Q £ ¡ £ fL; Rg$ である。ただし、定義されていない部分があってもかまわない。
- q_0 2 Q は初期状態。
- F µ Q は受理状態の集合。

チューリング機械 M の受理する言語とは、$§^¤$ の語のうち M が有限時間で停止し最終状態が受理状態になるような語の全体である。

Def. 5 (必ずしも有限停止しない) チューリング機械で受理される言語を帰納的可算集合 (recursively enumerable set) という。このとき受理されない語については有限時間で停止するかどうかは保証していない。すべての入力に対して有限時間で止まるようなチューリング機械で受理される言語を帰納的集合 (recursive set) と言う。

自然数上の関数の言葉に直せば、帰納的可算集合は帰納的部分関数に、帰納的集合は帰納的関数に対応すると見なせる。

帰納的可算な言語のクラスは非常に広く、文脈自由言語のクラスを真に含んでいる。

帰納的可算で帰納的でない言語のクラスには、与えられた語がその言語に属するか否かを決定できない言語がある。 $L(M)$ をそのような言語とすれば、与えられた入力語 w に対して、M が有限時間で止まれば w が $L(M)$ に属するかどうか判定できる。しかし、ここで入力語の中には M が停止しないものがあり、また M がまだ走っている間は $L(M)$ に属するのか、属さないのか言い当てることは出来ないので、この場合その語が $L(M)$ に属するか否か判定できない。

チューリング機械の定める関数　チューリング機械は自然数から自然数への関数ともみなせる。自然数はテープ上で通例、1進法で表される。つまり自然数 n は n 個並んだ 0 の列 0^n で表される。入力が k 変数ある時は、それらの 0 の列を 1 で区切って並べた列 $0^{n_1}10^{n_2}1\cdots10^{n_k}$ で表せばよい。

このような入力に対してチューリング機械が止まって (受理状態か否かにかかわらず) テープに残った記号列が 0^m であるときに $f(n_1; n_2; \ldots; n_k) = m$ と定義する。ここですべての自然数の k 組 $n_1; n_2; \ldots; n_k$ に対して値が定義されている必要はないとする。

「質問」　テープ上の記号列と自然数の間の対応を、上のような1進法ではなく通常の計算機のように2進法を採用すると、何か違いが生ずるか？

Def. 6 チューリング機械が常に有限時間で停止するとは限らないので、このように定めた自然数上の関数は一般に部分関数になる。これを計算可能な部分関数という。特に、すべての入力で有限停止するチューリング機械で定められる全域関数を計算可能と呼ぶ。

定理 2 チューリング機械で定義される計算可能部分関数は、帰納的部分関数のクラスに一致する。特に、計算可能関数は帰納的関数のクラスと一致する。

(Proof)　帰納的関数が計算可能関数であることを示すには、その最初のリストの中の関数がチューリング機械で実際に構成できることを示し、次に関数の合成と最小化がチューリング機械で実現できることを示せばよい。 ¤

自然数上のかけ算、 $n!$; $\lceil \log_2 n \rceil$; 2^n といった関数はすべて帰納的関数である。

チューリング機械に変更を加えたり、拡張を加えたりして次のような機械を考えることが出来るが、これらは計算能力の意味ではもとのチューリング機械と同等であることが知られている。

(1) テープを2方向無限にする。

(2) 多テープチューリング機械： k 本のテープと k 個のヘッドを持つ。

(3) 非決定性チューリング機械： 状態遷移関数が非決定的 (値が多値) のもの。(計算能力の意味で同等だが、計算時間は大きく違う。)

(4) 多次元チューリング機械：テープが2次元のマス目の並びになっているとき。 3次元以上でも定義できる。

Church-Türing の提唱　初等的算術関数をもとにして関数への合成、原始帰納法、最小化を有限回くりかえして得られる帰納的関数のクラス (Gödel-Kleene の一般帰納的関数) のほか、チューリング機械の意味で計算可能な関数のクラス (Türing)、Church による ¸-calculus 、Post システムなど、全く独立に提案されかつ全く形式化の異なるこれらのシステムが、関数のクラスとしてすべて一致することが知られている。このことから、「計算可能な関数」「それを計算するアルゴリズムの存在する関数」という直感的な概念とこの関数のクラスを同一視できる、というのが Church の提唱である。そもそも、「計算が可能」「人間が解くことが出来る」「アルゴリズムが存在する」といった言明は、それ自体は非数学的なものなので、これを証明することはできないが、上のように様々に独立に提唱された概念がすべて等価であることがこの考えの妥当性を保証すると考えられている。この意味で「アルゴリズム」という直感的概念は、「常に有限停止するチューリング機械」という概念に等価であるとみなせる。

(常に有限停止するとは限らない計算の手順は、通常「手続き」(procedure) と呼ぶ。)

1.3　決定不能問題

与えられた問題が Yes{No の答えを求める問題であるとき、判定問題と呼ばれる。以下では、判定問題だけを考えるが、一般に何かの最大値、最小値を求める問題は簡単に判定問題に帰着できるので、判定問題に限って一般性は失っていない。(最適値が満たす上限値と下限値の間で 2 分探索をくり返せばよいから。) また、ここでいう問題とは、パラメータによって定まる問題の例問 (instance) の集まりのことである。例問が有限個であるときは、後述のように問題が意味がなくなってしまうので、通常可算無限個の集まりである。例えば、「与えられた3つの正の整数 x, y, z がある三角形の3辺の長さになるか？」という問題では、 x, y, z のある3つの正整数の組がひとつの例問になる。例えば、、「3, 4, 8 は三角形の3辺の長さになるか？」がその例問 のひとつになる。そしてここの問題とは、 3つの正整数の組の全体のなす集合として表わされる。

これから、「与えられた (判定) 問題を解くアルゴリズムが存在するか (有限停止のチューリング機械が存在するか)」という形の命題を考える。アルゴリズムが存在するとき、その問題は決定可能であると呼ぶことにする。

これを少し精密に定義する。まず、問題のすべての例のパラメーターを符号化することにより、記号列、つまり語に置き換える。そして、判定問題の答えが Yes であるような語の全体からなる言語を考える。問題を解くアルゴリズムが存在するか否か決定することは、対応するこの言語を受理する有限停止のチューリング機械が存在するか決定することであり、それはつまりその言語が帰納的集合であるか判定することである。

Def. 7 判定問題 ¦ が決定可能 (decidable) であるとは、Yes の答えに対応する ¦ の例問 (instance) の符号化列の全体のなす集合 $¦_Y$ が、帰納的であることと定義する。決定可能でないとき、決定不能であるという。

具体例として、与えられた無向単純グラフにハミルトニアン閉路が存在するか、を判定する問題を考える。記号集合 § 上で無向単純グラフを符号化した語の全体を $¦_G$ とする。その中でハミルトン閉路の存在

するグラフの符号列の全体を $¦_{Ham}$ とする。ハミルトニアン閉路問題を解く「アルゴリズムが存在するか」という問題は、言語 $¦_{Ham}$ を受理する常に有限停止するチューリング機械が存在するかという問題に等しく、これは言いかえて、$¦_{Ham}$ が帰納的集合であるか判定することに同じになる。このとき、ハミルトン閉路問題は、任意の入力 $w 2 §^¤$ に対して $w 2 ¦_{Ham}$ となるかを判定する問題であるが、与えられた記号列がグラフの符号化列になっているかどうかは一般に容易に判定できるので、はじめから $w 2 ¦_G$ として考えることにする。

問題が無限の例 instance を持たないとき、その問題は常に決定可能になる。例えば「3次元のポアンカレ予想は正しいか？」という問題は、instance がひとつしかないので、奇妙に見えるが決定可能問題である。つまり、それを受理する有限停止チューリング機械が存在する。実際、任意の入力に対して Yes と出力して止まる M_1 と任意の入力に対して No と出力する M_2 のどちらかがその条件を満たす。（ただ、それがどちらであるかは、そこからは分からないが。）

帰納的、帰納的可算集合の性質

定理 3 帰納的言語の補集合は、帰納的である。

(Proof) 帰納的言語 L を認識する有限停止のチューリング機械 M があれば、その出力の Yes, No を反転して出力するチューリング機械が簡単に構成できる。¤

定理 4 言語 L とその補集合 $L^1 = §^¤ ¡ L$ が帰納的可算ならば L; L^1 は帰納的である。

(Proof) L, L^1 を受理するチューリング機械を M_1; M_2 とする。入力 w に対して、w を M_1; M_2 への入力として、M_1 が w を受理したとき Yes を、M_2 が受理したとき No を出力する機械を M とすれば、M は常に有限時間で停止し、かつそれが受理する言語は L になる。¤

このふたつの定理から、任意の L と L^1 に対して以下のうちのどれかが成立する。

(1) L と L^1 がともに帰納的である。

(2) L も L^1 もどちらも帰納的可算ではない。

(3) L と L^1 の一方が帰納的可算でかつ帰納的でなく、他方が帰納的可算でない。

対角線論法と対角言語　チューリング機械 (TM) 自身を § = f0; 1g 上で符号化したとして、チューリング機械 M の符号化列を < M > とする。(一般に、どのチューリング機械の符号列にもならない2進列がたくさんある。) 入力語 w 2 f0; 1g$^¤$ と組にして考えるときは、< M > の後に入力列 w を続けたものを < M;w > で表わす。

$(0 + 1)^¤$ の元を一列に並べて番号づけたとする。この中で i 番目の語を w_i とする。チューリング機械 TM のすべてを番号づけて M_1; M_2; M_3;::: と数え上げたとする。ある数え上げをひとつ固定して考えたとき、この番号 i をチューリング機械 M_i の指数または Gödel 数という。以下の議論は数え上げの仕方によらず成立する。要請される条件は、数 i が与えられたとき TM M_i を構成する手続きがあり、逆に M が与えられたとき、$M = M_i$ となる番号 i が計算できることである。

各 $i;j$ で $w_i \in L(M_j)$ であるとき $a(i;j) = 1$、そうでないとき $a(i;j) = 0$ として定まる無限の表 $(a(i;j) : i;j = 1;2;\cdots)$ を考える。これを使って言語 L_d を、$a(i;i) = 0$ のとき w_i は言語 L_d に属するとして定める。つまり

$$L_d = \{w_i \ (i = 1;2;\cdots) \ : \ a(i;i) = 0 \ i.e.; \ w_i \notin L(M_i)\}$$

定理 5 対角言語 L_d は、帰納的可算でない。$w_k \in L_d$ も $w_k \in \overline{L_d}$

(Proof) L_d がある TM M_k で受理される言語で $L_d = L(M_k)$ とする。同じ番号の語 w_k が L_d に属したとする。すると、$a(k;k) = 0$ で w_k は $L(M_k)$ に属さないことになり、矛盾。

語 w_k が L_d に属さないとする。すると、$a(k;k) = 1$ つまり w_k は $L(M_k)$ に属することになり再び $L_d = L(M_k)$ に矛盾する。排中律から、$L_d = L(M_k)$ という前提が間違っていたと分かる。 □

[質問] 対角線論法の本質は何だろうか。

Def. 8 $L_u = \{< M;w >: M$ は w を受理する $\}$ で定義される言語 L_u を万能言語 (universal language) という。

$(0 + 1)^*$ の中の語 w が TM M で受理されることと、語 $< M;w >$ が言語 L_u に属することとは同値になる。ここで簡単のため、チューリング機械の入力テープ記号が $\{0;1;B\}$ であるとしている。

定理 6 万能言語 L_u は帰納的可算である。

(Proof) チューリング機械 M が語 w を受理するときそのときに限り $< M;w >$ を受理するようなチューリング機械 M_u を構成してみせればよい。詳細略。 □

Def. 9 この定理より、L_u を受理するチューリング機械 M_u がある。この M_u を万能チューリング機械と呼ぶ。すなわち、この万能チューリング機械は、入力記号が 0, 1 である任意のチューリング機械を模倣できる。

定理 7 万能言語 L_u は帰納的でない。

(Proof) L_u を認識する有限停止のアルゴリズム A が存在したとする。すると対角言語 L_d の補集合 $\overline{L_d}$ を認識するアルゴリズムが次のように作れる。

$$w \ ! \ (i; \ w_i = w) \ ! \ < M_i; \ w_i > \ ! \ \boxed{A} \ ! \ Yes; \ No$$

任意の入力列 $w \in (0 + 1)^*$ に対して $w = w_i$ となる数 i は簡単に計算でき、また、i から $M = M_i$ となりうる TM M が構成できる。そこで $< M_i;w_i >$ をアルゴリズム A の入力にすれば、M_i が w_i を受理するときに限り、Yes を答え、そうでないときは No と答えるアルゴリズムができる。これは入力 w に対して $w \in \overline{L_d}$ を判定するアルゴリズムになっているが、$\overline{L_d}$ は帰納的でないので矛盾である。 □

これを言いかえると、問題「Turing 機械 M は語 w を受理するか？」は決定不能である。

Post の対応問題　別のタイプの決定不能問題の例を挙げる。Post の対応問題 (Post's Correspondence Problem, PCP) は、あるアルファベット集合上の語のふたつのリスト $A = (x_1; x_2; \cdots; x_k)$, $B = (y_1; y_2; \cdots; y_k)$ が与えられたとき、$x_{i_1} x_{i_2} \cdots x_{i_m} = y_{i_1} y_{i_2} \cdots y_{i_m}$ となる自然数の組 $i_1; i_2; \cdots; i_m$ が存在するかを判定する問題である。

例 2　$\S = \{0; 1\}$, $A = [x_1 = 1;\ x_2 = 10111;\ x_3 = 10]$, $B = [y_1 = 111;\ y_2 = 10;\ y_3 = 0]$ とすると $m = 4;\ i_1 = 2;\ i_2 = 1;\ i_3 = 1;\ i_4 = 3$ が共通列 $101111110 = x_2x_1x_1x_3 = y_2y_1y_1y_3$ を与えるので、解が存在する。。

例 3　次は解の存在しない例。 $A = [x_1 = 10;\ x_2 = 011;\ x_3 = 101]$, $B = [y_1 = 101;\ y_2 = 11;\ y_3 = 011]$

このとき、

定理 8　Post の対応問題は決定不能である。

プログラムの停止問題の形の対角線論法: 付録　「任意のプログラム P と入力データ D が与えられたとき、P に D を入力として与えたとき、プログラムが有限時間内に停止するか判定する」問題を考えよう。

これを判定できるプログラム T があったとする。つまり

$$T(P; D) = \begin{cases} \text{Yes} & \text{入力 D で P が止まるとき} \\ \text{No} & \text{入力 D で P が止まらないとき} \end{cases}$$

ここで T から次のようなプログラムを入力とするプログラム S が容易に作れる。

$$S(P) = \begin{cases} H(P;P) \text{ の出力が Yes ならば止まらない} \\ H(P;P) \text{ の出力が No ならば停止する} \end{cases}$$

ここでプログラム S の上で S 自身を入力としたとき、つまり S(S) は有限時間で停止するかどうか考えてみる。有限時間で停止するとすれば、定義から H(S;S) の出力が Yes である。このときプログラム S の定義から S(S) は停止しないことになる。矛盾。有限時間で停止しないとすれば、H(S;S) の出力は No になり、S(S) は有限時間で停止する。どちらも矛盾する。