計算論速習（2023 年度数学基礎論サマースクール）

東京都立大学 理 数理　鈴木 登志雄 [1]

目 次

1　チューリング マシン

1.1　帰納的関数（再帰的関数）

特に断りのない限り，関数への入力（以下の x_1 や x_2 など）は自然数であるとし，関数の値も自然数であるとする．

定義 **1.1** 以下の３パターンいずれかに該当する関数を初期関数という．

- 定数関数 $C_a(x_1, \ldots, x_n) = a$（ただし a は入力の値によらない自然数.）
- 後者関数 $S(x) = x + 1$（$x + 1$ のことを x' とも書く.）
- 射影関数 $U_i^n(x_1, \ldots, x_n) = x_i$

[1] revison a．修正箇所は赤字で表示.

定義 1.2 初期関数から出発し，以下の操作 1,2 を有限回（0 回も許す）適用して得られる関数を *primitive recursive function*（原始帰納的関数，原始再帰的関数）といい，以下の操作 1,2,3 を有限回（同上）適用して得られる関数を *recursive function*（帰納的関数，再帰的関数）という．

1. 合成　関数 g と $h_1, \ldots, h_m$ を用いて f を次のように定義する．

   $f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n))$

2. 原始帰納法（原始再帰法）　定数 a と関数 h を用いて f を次のように定義する．$\begin{cases} f(0) = a & \text{（定数）} \\ f(y') = h(y, f(y)) \end{cases}$

   関数 g と h を用いて f を次のように定義する（ただし $n \geq 2$ の場合）．

   $$\begin{cases} f(0, x_2, \ldots, x_n) = g(x_2, \ldots, x_n) \\ f(y', x_2, \ldots, x_n) = h(y, f(y, x_2, \ldots, x_n), x_2, \ldots, x_n) \end{cases}$$

3. 全域最小化　各々の自然数 $x_1, \ldots, x_n$ に対して自然数 y が存在して $g(x_1, \ldots, x_n, y) = 0$ となるとする（このとき，g は正則性条件をみたすという）．このような g を用いて f を次のように定義する．$f(x_1, \ldots, x_n) = \min\{y : g(x_1, \ldots, x_n, y) = 0\}$

例題 1.1 自然数の加算 $x + y$，乗算 xy，階乗 $x!$ が原始帰納的関数であることを示せ．

解

$x + y$
$$\begin{cases} x + 0 & = x \\ x + y' & = (x + y)' \end{cases}$$

xy
$$\begin{cases} x0 & = 0 \\ xy' & = xy + x \end{cases}$$

$x!$
$$\begin{cases} 0! & = 1 \\ (x')! & = x'(x!) \end{cases}$$

□

問 1.1 以下の関数が原始帰納的であることを示せ．

(1) 指数（0^0 は 1 とする）

$$x^y$$

(2) 符号

$$\mathrm{sgn}(x) = \begin{cases} 0 & \text{If } x = 0 \\ 1 & \text{それ以外} \end{cases}$$

(3) ゼロかどうかの判定

$$\mathrm{is\_zero}(x) = \begin{cases} 1 & \text{If } x = 0 \\ 0 & \text{それ以外} \end{cases}$$

(4) 前者関数

$$\mathrm{pd}(x) = \begin{cases} 0 & \text{If } x = 0 \\ x - 1 & \text{それ以外} \end{cases}$$

(5) 固有差（引き算もどき）

$$x \dot{-} y = \begin{cases} 0 & \text{If } x \leq y \\ x - y & \text{それ以外} \end{cases}$$

(6) ２で割った剰余（0, 1 で表す）　$x \mod 2$

1.2　述語と制御構造

高校数学で条件とよんだものを大学数学では *predicate*（述語）ともいう．真を 1，偽を 0 で表し，与えられた k 変数述語 P を $\mathbb{N}^k$ から $\{0,1\}$ への関数とみなしたものを，元の述語の *characteristic function*（特性関数）という．本稿では χ_P で表す．また $A \subseteq \mathbb{N}$ に対し，述語「$n \in A$」の特性関数を A の特性関数という．述語あるいは集合の特性関数が（原始）帰納的であるとき，元の述語あるいは集合は（原始）帰納的であるという．たとえば，奇数全体の集合は原始帰納的集合である（特性関数は $x \mod 2$ だから）．

例題 1.2 述語 $P(x_1,\ldots,x_n)$ が（原始）帰納的述語であるとき，その否定 $\mathrm{not}\ P(x_1,\ldots,x_n)$ も（原始）帰納的であることを示せ．

略解　$\chi_{\text{not } P}(x_1,\ldots,x_n) = \mathrm{is\_zero}(\chi_P(x_1,\ldots,x_n))$．（別解　$1 \dot{-} \chi_P(\vec{x})$）□

問 1.2 述語 $P_1(x_1,\ldots,x_n)$, $P_2(x_1,\ldots,x_n)$ が（原始）帰納的述語，関数 $f_1(x_1,\ldots,x_n)$, $f_2(x_1,\ldots,x_n)$ が（原始）帰納的関数であるとき，以下の述語および関数が（原始）帰納的であることを示せ．

(1) 連言　$P_1(x_1,\ldots,x_n)$ and $P_2(x_1,\ldots,x_n)$

(2) 選言　$P_1(x_1,\ldots,x_n)$ or $P_2(x_1,\ldots,x_n)$

(3) 場合分け　$f_3(x_1,\ldots,x_n) = \begin{cases} f_1(x_1,\ldots,x_n) & \text{If } P_1(x_1,\ldots,x_n) \text{ holds} \\ f_2(x_1,\ldots,x_n) & \text{それ以外} \end{cases}$

次に原始帰納法と全域最小化が，プログラミング言語の制御構造とどのように関係しているかをみよう（より詳しくは [鹿島 2008] 参照）．

例 1.3 原始帰納法で定義された関数 $\begin{cases} f(0) = a & \text{（定数）} \\ f(y') = h(y, f(y)) \end{cases}$ は締め切りつきのループを用いて以下のアルゴリズムで計算できる．

```
int f ( int x ){ int y, z = a;
  for ( y = 0; y < x; ++y ){ z = h(y, z); }
  return z;
}
```

例 1.4 全域最小化で定義された関数 $f(x) = \min\{y : g(x,y) = 0\}$ は，停止が保証された（ただし締め切りなしの）ループを用いて計算できる．

```
int f ( int x ){ int y = 0, z;
  z = g( x, y );
  while ( z != 0 ){ ++y; z = g( x, y ); }
  return y;
}
```

1.3　チューリング マシンとチューリング計算可能関数

集合 A のある部分集合から集合 B への関数を，A から B への部分関数という．とくに定義域が A と一致する場合は A から B への全域的関数という．

定義 **1.3** $M = \langle Q, \Sigma, \delta, q_0 \rangle$ が以下をみたすとき，M をチューリング マシンという．

- Q は有限集合である．Q の元を（有限）状態という．
- $\Sigma = \{1, B\}$．B を空白文字，Σ をアルファベットという．$Q \cap \Sigma = \emptyset$.
- q_0 は Q の要素であり，初期状態とよばれる．
- δ は $Q \times \Sigma$ から $\Sigma \times \{L, C, R\} \times Q$ への部分関数（動作関数）．

以下ではチューリング マシンをしばしば Tm と略す．

定義 **1.4** $M = \langle Q, \Sigma, \delta, q_0 \rangle$ が Tm であるとする．

1. 文字列 $a_1 \cdots a_n : q$ が以下をみたすとき，この文字列を M の時点表示という．「Σ の要素の有限列（長さ 1 以上）があって，その 1 か所だけに下線をつけたものが $a_1 \cdots a_n$ になっている」この，下線のついた文字をヘッドが指す文字という．q は Q の要素であり，現在の状態とよばれる．
2. 時点表示 $a_1 \cdots a_n : q$ と $Ba_1 \cdots a_n : q$ と $a_1 \cdots a_n B : q$ は同値であるとする．すなわち，$a_1 \cdots a_n$ の部分の左右に有限個の空白記号を付け加えたり，削除したりして得られる時点表示は，元の時点表示と同値であるとする．

上記二つの定義について，非公式な（本音の）解釈は以下の通りである．Tm のハードウェアは，本体と，マス目（セル）に区切られて左右に無限に伸びたテープ，ヘッドからなる．時間は離散的に進む．

チューリング マシンはこんな部品でできている．

マス目に区切られたテープ

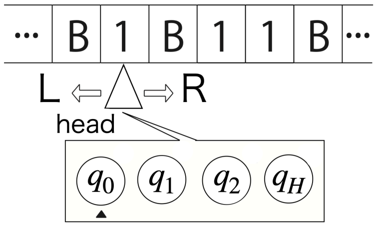
使える文字はＢ（空白記号）と 指定された有限個（この例では１だけ）．B 以外はテープ上に有限回出現する．

ヘッド．

各時点で特定のマス目を指す．読み出しと書き込みができる． 左または右に１マスずつ動ける．動かなくてもいい．

状態．

有限個の状態をもつ（この例では４個）．各時点で「現在の状態」がある（▲）．q_0 は初期状態．



プログラム本体．

下記参照．

プログラム本体の例（この６行で一つのプログラム）

状態が q_0 でヘッドが指す文字が B なら１マス右へ行き状態を q_1 にせよ．

状態が q_0 でヘッドが指す文字が 1 なら１マス右へ行き状態を q_1 にせよ．

状態が q_1 でヘッドが指す文字が B なら１マス右へ行け．

状態が q_1 でヘッドが指す文字が 1 なら１マス右へ行き状態を q_2 にせよ．

状態が q_2 でヘッドが指す文字が B なら１マス右へ行き状態を q_1 にせよ．

状態が q_2 でヘッドが指す文字が 1 なら B に書き換え，状態を q_H にせよ．

$M = \langle Q, \Sigma, \delta, q_0 \rangle$ が Tm のとき動作関数 δ がプログラム本体の役割を果たす． $\delta(q,a) = \langle b, L, r \rangle$ という等式を以下の命令と解釈する．

```
if ( 現在の状態 == q かつ ヘッドが指す文字 == a ) {
    ヘッドが指す文字を b に書き換え，
    ヘッドを一つ左のセルに移動し，
    現在の状態を r に変更せよ
}
```

上記の L の部分が R, C のときはそれぞれ，上記の「左のセル」の部分を「右のセル」，「現在のセル」で置き換える．「$\delta(q,a) = \langle b, X, r \rangle$」の形の命令を１回実行することを１ステップという．

時点表示は，ある瞬間のテープの様子とヘッドの位置と状態を表す．テープ文字が書いていないところはすべて空白文字 B で埋め尽くされているものと解釈する．たとえば，上記枠内イラストは，時点表示 $B\underline{1}B11B : q_0$ で表せる．また，時点表示 $BBB\underline{1}B11BBB : q_0$ で表すこともできる．

計算開始時点の状態は初期状態 q_0 である．ヘッドが指す文字を a として，$\langle q, a\rangle$ が δ の定義域に属しているなら 1 ステップの処理を行う．この処理を可能な限り続ける．ある瞬間に $\langle q, a\rangle$ が δ の定義域に属さなくなったとき，M は停止する．M の計算がいつまでも停止しない場合（発散）もあり得る．建前上は，物理的な時間の概念を使わずに計算過程を文字列として定義する．

定義 1.5 $M = \langle Q, \Sigma, \delta, q_0\rangle$ が Tm であるとする．

1. 時点表示 $\alpha = a_1 \cdots a_n : q$, $\beta = b_1 \cdots b_m : q'$ に対して以下の条件が成り立つとき，「α は M の 1 ステップによって β に移行する」という．

   条件:α においてヘッドが指す文字を a_s とする．このとき，順序対 $\langle q, a_s\rangle$ が δ の定義域に属し，$\delta(q, a_s) = \langle b, X, r\rangle$ とおくと，$r = q'$ かつ，以下のどれかが成り立つ．

   - $X = C$ かつ，α において $\underline{a_s}$ となっている部分を $\underline{b}$ に書き換え q を r に書き換えて得られる文字列が β と同値である．
   - $X = L$ かつ，$s = 1$ かつ，α において $\underline{a_1}$ となっている部分を $\underline{B}b$ に書き換え q を r に書き換えて得られる文字列が β と同値である．
   - $X = L$ かつ，$s \geq 2$ かつ，α において $a_{s-1}\underline{a_s}$ となっている部分を $\underline{a_{s-1}}b$ に書き換え q を r に書き換えて得られる文字列が β と同値である．
   - $X = R$ かつ，$s = n$ かつ，α において $\underline{a_n}$ となっている部分を $b\underline{B}$ に書き換え q を r に書き換えて得られる文字列が β と同値である．
   - $X = R$ かつ，$s < n$ かつ，α において $\underline{a_s}a_{s+1}$ となっている部分を $b\underline{a_{s+1}}$ に書き換え q を r に書き換えて得られる文字列が β と同値である．

2. 時点表示有限個の列 $\alpha_0, \alpha_1, \ldots, \alpha_k$ において以下二つの条件が成り立つとき，M は α_k で停止するといい，列 $\alpha_0, \alpha_1, \ldots, \alpha_k$ を M の k ステップの計算（過程）という．

   - 各 $j < k$ に対して，α_j は M の 1 ステップによって α_{j+1} に移行する．
   - α_k が $\cdots\underline{a}\cdots : p$（ただし $\langle p, a\rangle$ は δ の定義域に属さない）の形をしている．

自然数 n の 1 進コードとは，記号 1 を $n+1$ 個並べたものとする．たとえば 0 のコードは 1，3 のコードは 1111 である．

定義 1.6 1. $Tm\,M$ が関数 $f:\mathbb{N}^n \to \mathbb{N}$ を計算するとは，任意の $\langle x_1,\ldots,x_n\rangle \in \mathbb{N}^n$ に対して以下が成り立つことをいう（ただし q_0 は M の初期状態，q_H は停止状態とする）．M の計算過程 $\alpha_0,\alpha_1,\ldots,\alpha_k$ が存在して α_0 は

$$B^* \quad \underbrace{1\cdots1}_{x_1+1\,個} \quad B \quad \underbrace{1\cdots1}_{x_2+1\,個} \quad B\cdots B \quad \underbrace{1\cdots1}_{x_n+1\,個} \quad \underline{B}B^*:q_0 \tag{1}$$

であり，かつ，α_k は以下の形をしている．ただし B^* は B が 0 個以上並んだものを表す．

$$B^* \quad \underbrace{1\cdots1}_{x_1+1\,個} \quad B \quad \underbrace{1\cdots1}_{x_2+1\,個} \quad B\cdots B \quad \underbrace{1\cdots1}_{x_n+1\,個} \quad B \quad \underbrace{1\cdots1}_{f(x_1,\ldots,x_n)+1\,個} \quad \underline{B}B^*:q_H$$

2. 関数 $f:\mathbb{N}^n \to \mathbb{N}$ がチューリング計算可能であるとは，f を計算する Tm が存在することをいう．

なお，上記 (1) を $\overline{\langle x_1,\cdots,x_n\rangle}/q_0$ で表すことがある．

1.4 帰納的関数を計算するチューリング マシン

定理 1.7 帰納的関数はチューリング計算可能である．

上記定理の証明を，何段階かに分けて以下で行う．本節では，とくに断りのない限り関数とは全域的関数を表すものとする．

問 1.3 次のような Tm が存在することを示せ．ヘッドを現在位置から 1 マス以上左へ動かして，最初に出会う $B1$ の 1 のところで停まる．もし $B1$ に出会わなければいつまでも左へヘッドを進ませて停まらない．

考え方 § 1.3 の囲み記事「チューリング マシンはこんな部品でできている」と同様．

例題 1.5 次のような Tm が存在することを示せ．どんな自然数 x に対しても，時点表示

$$Ba_1\cdots a_nB \quad \underbrace{1\cdots1}_{x+1\,個} \quad \underline{B}:q_0$$

から計算を開始すると有限ステップの後，以下の時点表示で停まる．

$$Ba_1 \cdots a_n B \quad \underbrace{1 \cdots 1}_{x+1\text{個}} \quad B \quad \underbrace{1 \cdots 1}_{x+1\text{個}} \quad \underline{B} : q_H$$

略解　作業の流れは以下の通りである．まず 1 のブロックの左端に目印を付ける．次に 1 のブロックの左端から順番に，1 を右側の新ブロックへ移送する．配送済みの記録として，1 があった場所は B に変えておく．これを繰り返して，元々の 1 のブロックを右側の新ブロックの場所へすっかりコピーする．最後に終了処理として，元のブロックを復元する．その際，最初に付けた左端の目印を利用する．では，この一連の作業をチューリングマシンとして実現できることを見ていこう．

状態 q_0 からの動作：問 1.3 と同様にしてチューリングマシンを設計し，まず次のような動作をさせる．左へ行って最初の $xB1$ を探す（上の例では $x = a_n$ であり，その値は 1 かもしれないし B かもしれない）．より正確に言えば，この動作を行う前にヘッドがあった位置よりも真に左側に x があり，そこから $xB1$ となっているもののうち，最も右にあるものを探す．みつからなければ永久に左へ行く（以下ではこの種の断り書きは省略する）．みつかれば $xB1$ を $1B1$ に書き換え，$1B1$ の右側の 1 で停まって状態 q_1 になる．このとき x が 1 であったかどうかは状態によって記憶する．つまり x が 1 なら状態 $q_1^{(1)}$，B なら状態 $q_1^{(B)}$ とする．このときの状態に応じて，後述の（∗）の時点まで，「状態を q_i にする」という部分は厳密には $q_i^{(1)}$ か $q_i^{(B)}$ のいずれかにするのであるが，煩雑なので省略する．また，q_0 から q_1 に至る間にいくつかの状態をとるが，それらについての説明も省略する．

たとえば最初の時点表示が

$$Ba_1 \cdots a_n B111\underline{B} : q_0$$

であれば，ここまでの動作で以下の時点表示に至る．

$$Ba_1 \cdots 1B\underline{1}11B : q_1$$

状態 q_1 からの動作：ヘッドが指す文字を B に書き換え，この B を 1 番目として右へ行って 3 番目の B をみつけ，それを 1 に書き換える．次に左へ行って最初に出会う B をみつけ，そのさらに 1 マス左へ行く．そこが 1 のときは状態 q_2，B のときは状態 q_3 になる．

上記の例の続きは，

$$Ba_1 \cdots 1BB11B\underline{1}$$

を経て以下の時点表示に至る．

$$Ba_1 \cdots 1BB1\underline{1}B1 : q_2$$

状態 q_2 からの動作：左へ行って最初に出会う $B1$ をみつけ（もともとヘッドが 1 を指していてその左が B の場合は，その $B1$），その $B1$ の 1 で停まり，状態を q_1 に戻す．

上記の例の続きは，まず

$$Ba_1 \cdots 1BB\underline{1}1B1 : q_1$$

となる．ここで q_1 からの動作を行うので，上記と同様にして

$$Ba_1 \cdots 1BBB1B1\underline{1}$$

を経て

$$Ba_1 \cdots 1BBB\underline{1}B11 : q_2$$

となる．ここで q_2 からの動作を行うので，上記と同様に，

$$Ba_1 \cdots 1BBB\underline{1}B11 : q_1$$

となり，続いて q_1 からの動作を行うので，

$$Ba_1 \cdots 1BBBBB11\underline{1}$$

を経て以下の時点表示に至る．状態が q_3 になることに注意．

$$Ba_1 \cdots 1BBB\underline{B}B111 : q_3$$

状態 q_3 からの動作：ヘッドが指す文字を 1 に書き換える．左へ行って最初の $1B$ をみつける．第 1 段落で出会った x が B であれば（これは状態によって記憶している）$1B$ を BB に書き換える（∗）．これで，最初に a_nB だった箇所が元通りになる．a_n の右隣の B から 1 マス右へ行く．そこが 1 でなければ，1 に出会うまで，B を 1 に書き換えながら右へ進む．1 に出会ったら右へ進み，２回目に出会う B で，状態を q_H にして停止する．

上記の例の続きは，

$$Ba_1 \cdots \underline{a_n}BBB1B111$$

さらに

$$Ba_1 \cdots a_nB111B111\underline{B} : q_H \downarrow$$

となる．ここで計算が終わる． □

さて，n を正の整数とするとき，アルファベットを $\Sigma_n = \{B, 0, 1, \ldots, n\}$ に拡張したＴｍを考える．この拡張マシンに基づく計算可能性を容易に定義できる．

補題 1.8 M_2 は，アルファベットを $\Sigma_1 = \{B, 0, 1\}$ に拡張したＴｍであるとする．このとき，普通の（すなわち，アルファベットが $\Sigma = \{B, 1\}$ である）Ｔｍ M_1 が存在して，M_1 によって M_2 を模倣できる．ただし，Σ_1 の文字を以下の規則で Σ の文字列に変換して模倣を行うものとする：$B \mapsto BB, 0 \mapsto B1, 1 \mapsto 1B$.

証明の概略　拡張マシン M_2 の各状態 q に対して，文字コード（たとえば B のコードは BB）の左半分を読んでいる状態 q' と右半分を読んでいる状態 q'' を導入して M_1 を構成する． $\square$

同様の議論により，以下を得る．

補題 1.9 n を正の整数とする．アルファベットを $\Sigma_n = \{B, 0, 1, \ldots, n\}$ に拡張したＴｍで計算可能な関数は，アルファベットが $\Sigma = \{B, 1\}$ である普通のＴｍによって計算可能である（ただし，必要に応じて補題 1.8 と同様の文字コードを設定する）．

次に多テープ多ヘッドＴｍを考える．テープは有限本あるものとする．各テープごとに一つずつヘッドを割り当てる．状態はすべてのヘッドで共有する．すべてのヘッドがテープ文字を読み込み，それらの結果に応じて，各ヘッドが書き込みと移動を行って次の状態を指定するまでを１ステップとする．たとえばヘッド A とヘッド B が読み込んだ文字が何であるかによってヘッド C の動作を変えることも許す．

補題 1.10 n, m を正の整数とする．アルファベットを $\Sigma_n = \{B, 0, 1, \ldots, n\}$ に拡張し，さらにテープの本数を m 本にに拡張したＴｍで計算可能な関数は，アルファベットが $\Sigma = \{B, 1\}$ でテープの本数が 1 本である普通のＴｍによって計算可能である．

証明の概略　チューリング計算可能性の概念を不変にしたまま，Ｔｍのアルファベットを任意の有限個にしてもよいので，句読点や区切り記号をアルファベットに付け加えてもよい．また，ヘッドが指す B を表す文字 $\underline{B}$ や，ヘッドが指す 1 を表す文字 $\underline{1}$ をアルファベットに付け加えてもよい．このようにして，複数のテープについての時点表示を１本のテープの中に表現することができる． $\square$

補題 1.11 ・定数関数 $C_a(x_1, \ldots, x_n) = a$ はチューリング計算可能である．
・後者関数 $S(x) = x'$ はチューリング計算可能である．
・射影関数 $U_i^n(x_1, \ldots, x_n) = x_i$ はチューリング計算可能である．

証明の概略　具体的にＴｍを構成することにより示せばよい． $\square$

補題 1.12 チューリング計算可能な関数の合成はチューリング計算可能である．

証明の概略　例として $g_1(x_1), g_2(x_1), h(x_1, x_2)$ がチューリング計算可能であるとして，合成関数 $f(x) = h(g_1(x), g_2(x))$ について考察する．４テープＴｍによって以下のような計算を行う．入力 x のコードは第１テープに書

くものとする．第２テープと第３テープに x のコードをコピーする．第２テープ上で $g_1(x)$ を計算し，第３テープ上で $g_2(x)$ を計算する．第４テープに $\langle g_1(x), g_2(x)\rangle$ のコードを書き込み，$h(g_1(x), g_2(x))$ を計算する．その結果を第１テープにおける x の右隣に（１マス空白をはさんで）コピーする．この４テープＴｍは $f(x)$ を計算する．よって補題 1.10 により，$f(x)$ はチューリング計算可能である． □

補題 1.12 と同様にして，以下二つの補題を証明できる．

補題 **1.13** チューリング計算可能な関数のみを用いて原始帰納法によって定義された関数はチューリング計算可能である．

補題 **1.14** チューリング計算可能な関数のみを用いて 全域最小化によって定義された関数はチューリング計算可能である．

定理 **1.7** の証明の概略　補題 1.11，1.9，1.10，1.12，1.13，1.14 と帰納法による． □

2　停止問題

2.1　機械の番号付け

以下のようにして Tm の列 $\{M_e\}_e$ を定めることを，インフォーマルには

「すべての Tm に番号（インデックス）を付けて $\{M_e\}_e$ とする」
「$\{M_e\}_e$ を Tm の番号付け（enumeration）とする」

などと言うことがある。

1. q_{110} のように q_{ビット列} の形の語で Tm の個々の状態を表すことにしても一般性を失わない．
2. 文字の有限集合（アルファベット）$\Gamma = \{q, 0, 1, B, \dots\}$ を適当に定め，個々の Tm はこのアルファベット上の語とみなす。つまり C 言語や Python のソースコードのように，Tm のソースコードを考える．
3. 計算可能な全単射 $f : \{0,1\}^* \to \Gamma^*$ で，逆関数も計算可能なものを一つ固定する．定義域は $\mathbb{N}$ と同一視する．
4. 補欠役の Tm N を一つ固定する．
5. 自然数 e が与えられたとき，もし $f(e)$ が Tm のソースコードになっていればその Tm を M_e とする．そうでない場合は補欠 N を M_e とする．

M_e によって自然に定まる一変数部分帰納的関数を考えることにより，上記によって一変数部分帰納的関数の enumeration $\{\varphi_e(x)\}_e$ を得る．インフォーマルな議論では M_e と e の区別があいまいな書き方をすることがある．

2.2　クリーネの標準形定理

以下では，話を簡単にするため，「時点表示 $\overline{\langle x_1, \cdots, x_n \rangle}/q_0$ から M の計算を開始すると，有限回の動作の後に $\overline{\langle x_1, \cdots, x_n, z \rangle}/q_H$ （ただし z は自然数で，q_H は M の停止状態）の形の時点表示に到って，そこで M が停止する」ということを「M は入力 $\langle x_1, \cdots, x_n \rangle$ に対して停止する」といおう．また，上の z を「入力 $\langle x_1, \cdots, x_n \rangle$ に対する M の出力」といおう．

補題 2.1 $M = \langle Q, \Sigma, \delta, q_0 \rangle$ はチューリング マシン（ただし $\Sigma = \{1, B\}$）であり，n, k は正の整数であり，かつ $x_1, \cdots, x_n$ は自然数であるとする．

(1) M が入力 $\langle x_1, \cdots, x_n \rangle$ に対して停止するための必要十分条件は，以下の性質 1 を持つような文字列の列 $\alpha_0, \alpha_1, \cdots, \alpha_k$ が存在することである（各々の α_j が文字列）．

（性質 1）:「α_0 は　(0 個以上の B) $\overline{\langle x_1, \cdots, x_n \rangle}/q_0$ (0 個以上の B)　の形をしており，α_k は
(0 個以上の B) $\overline{\langle x_1, \cdots, x_n, z \rangle}/q_H$ (0 個以上の B)　の形をしており，かつ $\alpha_0, \alpha_1, \ldots, \alpha_k$　は M の計算である．」

(2) $\alpha_0, \alpha_1, \cdots$ および α_k は任意の文字列であるとする．これらの文字列からなる列

$$\alpha_0, \alpha_1, \cdots, \alpha_k \tag{2}$$

が上記の性質 1 を持つための必要十分条件は，(2) が以下のアルゴリズムによって受理されることである．なお，このアルゴリズムは（C 言語とは少し違う）擬似コードで書いてある．

---

begin (アルゴリズム $T_n{}'$)
input チューリング マシン（のプログラム）M,
　　自然数 $x_1, \cdots, x_n$,
　　文字列の列 $y = (\alpha_0, \alpha_1, \cdots, \alpha_k)$.

if α_0 が　(0 個以上の B) $\overline{\langle x_1, \cdots, x_n \rangle}/q_0$ (0 個以上の B)　の形でない
then reject /* 入力を拒否して終了する */**end-if**

if α_k が　(0 個以上の B) $\overline{\langle x_1, \cdots, x_n, z \rangle}/q_H$ (0 個以上の B)
（ただし z は自然数であり，q_H は M の停止状態）　の形でない
then reject **end-if**

for $j = 0, \cdots, k-1$
　if M の 1 ステップの動作によって時点表示 α_j から時点表示 α_{j+1} へ移行しない
　then reject **end-if**
end-for
/* 以上すべてのテストに合格したら */
accept /* 入力を受理して終了する */
end

補題 2.1 の証明は容易であり，省略する．

補題 2.2 (1) 原始帰納的関数 U をうまく定めると，任意の正の整数 n に対して原始帰納的関数 T_n を定め，以下が成り立つようにできる．n 変数の入力をもつ $\mathrm{Tm}\ M$ であって，任意の入力 $\langle x_1, \cdots, x_n\rangle \in \mathbb{N}^n$ に対して停止するものと，任意の自然数 $x_1, \cdots, x_n$ に対して以下が成り立つ．

$$M(x_1, \cdots, x_n) = U(\min\{y : T_n(M, x_1, \cdots, x_n, y) = 0\})$$

ただし，左辺は入力 $\langle x_1, \cdots, x_n\rangle$ に対する M の出力を表す．また，右辺の M は（あらかじめ決めた文字コードで）M を自然数に変換したものを表す．

(2) f が n 変数の関数で，$\mathrm{Tm}\ M$ が f を計算するとき以下が成り立つ．ただし U と T_n は (1) のものである．

$$f(x_1, \cdots, x_n) = U(\min\{y : T_n(M, x_1, \cdots, x_n, y) = 0\})$$

とくに n 変数の帰納的関数 f に対して，f を計算する $\mathrm{Tm}\ M$ をとると上記が成り立つ．

(3) 全域的なチューリング計算可能関数は帰納的関数である．

証明の概略　　(1) 補題 2.1 において，$y = \langle \alpha_0, \alpha_1, \cdots, \alpha_k\rangle$ から z を読み出す関数を U とおく．関数 U は y から α_k を読み出し，さらに α_k から z を読み出す関数であり，n と無関係なアルゴリズムによって計算できる．

M は題意の条件をみたす Tm，$x_1, \cdots, x_n$ は任意の自然数とする．補題 2.1 により，M が入力 $\langle x_1, \cdots, x_n\rangle$ に対して停止するための必要十分条件は，

$$\exists y\ T_n{}' \text{ は } \langle M, x_1, \cdots, x_n, y\rangle \text{ を受理する}$$

である．そして，このような y に対して，$z = U(y)$ が成り立つ．さて，適当な文字コードを定めておけば，チューリングマシンのプログラムを書くのに必要な記号はすべて 2 進数だと思ってよく，結局，上記の M や y も自然数だと思ってかまわない[2]．そこで関数 T_n を以下のように定める．

$$T_n(M, x_1, \cdots, x_n, y) = \begin{cases} 0 & (T_n{}' \text{ が } (M, x_1, \cdots, x_n, y) \text{ を受理するとき}) \\ 1 & (\text{そうでないとき}) \end{cases}$$

アルゴリズム $T_n{}'$ において使われているループは for $j = 0, \cdots, n$ の形のものだけであり，while ループが用いられていないことに注意すると，関数 T_n が原始帰納的関数であることを証明できる．また，同様の理由により，U も原始帰納的関数である．

以上により (1) が成り立つ．

(2) 「Tm M が f を計算する」の定義と上記 (1) により (2) が成り立つ．

(3) 「f がチューリング計算可能」の定義と上記 (1) により (3) が成り立つ．
□

[2] 本当は M や y そのものと，M や y に対応する自然数（コード）を区別すべきである．厳密な議論は専門書を参照．

定理 1.7 と補題 2.2 により，次の定理を得る．

定理 2.3 全域的なチューリング計算可能関数全体の集合と全域的な帰納的関数全体の集合は一致する．

部分関数に対しても全く同様の議論が成り立つ．

定義 2.4 (1) $\mathbb{N}^n$ の部分集合から $\mathbb{N}$ への関数を（n 変数の）部分関数という．

(2) 帰納的関数の定義において，(VI) の g の正則性条件を取り去った条件を考え，それを「(VI $'$) 最小化演算」と呼ぼう[3]．詳しくは以下の通り．まず，自然数の集合の最小値をとる演算 min を修正して，μ 作用素を導入する．

自然数 $x_1, \ldots, x_n$ に対して，ある自然数 k があって

- すべての自然数 $z \leq k$ に対して $(x_1, \ldots, x_n, z)$ が g の定義域に属し，
- すべての自然数 $z < k$ に対して $g(x_1, \ldots, x_n, z) \neq 0$，かつ
- $g(x_1, \ldots, x_n, k) = 0$

となるとき，$\mu y\ (g(x_1, \ldots, x_n, y) = 0)$ の値はこの k と定める．このような k がないときは $\mu y\ (g(x_1, \ldots, x_n, y) = 0)$ は未定義とする．

最小化

与えられた関数 g を用いて f を次のように定義する．$f(x_1, \ldots, x_n) = \mu y\ (g(x_1, \ldots, x_n, y) = 0)$

(I) 定数関数，(II) 後者関数，(III) 射影関数を初期関数として，(IV) 合成演算，(V) 原始帰納法演算および (VI $'$) 最小化演算を有限回（0 回も可）施すことによって得られる関数を部分帰納的関数という．

(3) n 変数の部分関数 f がチューリング計算可能な部分関数であるとは，あるチューリング マシン M が存在して，任意の n 組 $\langle x_1, \cdots, x_n \rangle \in \mathrm{N}^n$ に対して以下のふたつの条件が成り立つことをいう．

- $\langle x_1, \cdots, x_n \rangle$ が f の定義域に属するとき，時点表示 $\overline{\langle x_1, \cdots, x_n \rangle}/q_0$ から M の計算を開始すると，有限回の動作の後に時点表示 $\overline{\langle x_1, \cdots, x_n, f(x_1, \cdots, x_n) \rangle}/q_H$ に到って，そこで M は停止する．ここで，q_0 は M の初期状態，q_H は M の停止状態である．
- $\langle x_1, \cdots, x_n \rangle$ が f の定義域に属さないとき，時点表示 $\overline{\langle x_1, \cdots, x_n \rangle}/q_0$ から M の計算を開始すると，M は停止しない．

[3] 最小化の定義は文献によって微妙に違う．[鹿島 2008, pp.101–104] 参照．

定理 **2.5** (1) チューリング計算可能な部分関数全体の集合と部分帰納的関数全体の集合は一致する．

(2) （クリーネの標準形定理）n 変数の任意の部分帰納的関数 f に対して

$$f(x_1, \cdots, x_n) \simeq U(\ \mu y(T_n(M, x_1, \cdots, x_n, y) = 0)\ ) \tag{3}$$

が成り立つ．ここで，M は f を計算するチューリング マシン（のプログラムを自然数に変換したもの）．また，$\simeq$ は両辺の定義域が一致して個々の関数値も一致することを表す．

2.3 パラメータ定理

$\{\varphi_e\}$ を部分帰納的関数の enumeration とする．

定理 **2.6** (Enumeration theorem) 各自然数 k に対し，関数

$$(e, x_1, \ldots, x_k) \mapsto \varphi_e(x_1, \ldots, x_k)$$

は $e, x_1, \ldots, x_k$ の部分帰納的関数である．

定理 **2.7** (Parameter theorem, s-m-n theorem) $F(x_1, \ldots, x_k, y_1, \ldots, y_m)$ が部分帰納的関数のとき，以下を満たす全域的帰納的関数 S が存在する．

任意の $x_1, \ldots, x_k, y_1, \ldots, y_m$ に対し，

$$\varphi_e(x_1, \ldots, x_k) \simeq F(x_1, \ldots, x_k, y_1, \ldots, y_m),$$

ただし $e = S(y_1, \ldots, y_m)$．

2.4　停止問題

以下では，チューリング計算可能関数のことを単に計算可能関数とよぼう．チューリング マシン（のプログラムを自然数に変換したもの）M と自然数 x を入力として受け取り，M が入力 x に対して停止するかを判定する問題を（チューリング マシンの）停止問題という．

定理 2.8 停止問題は計算可能ではない．より詳しく言えば，以下によって定義される関数 f は計算可能ではない．

$$f(M,x) = \begin{cases} 1 & (M \text{ が入力 } x \text{ に対して停止するとき}) \\ 0 & \text{（そうでないとき）} \end{cases}$$

証明の概略　　背理法の仮定として上記の関数 f が計算可能だとする．すると任意の入力 x に対して次のように動作する，チューリング マシン（のプログラムを自然数に変換したもの）M がある．

$$f(x,x) = 1 \text{ のとき，} M \text{ は入力 } x \text{ に対して停止しない．}$$
$$f(x,x) = 0 \text{ のとき，} M \text{ は入力 } x \text{ に対して } 1 \text{ を出力して停止する．}$$

すると

$$f(M,M) = 1 \text{ のとき，} M \text{ は入力 } M \text{ に対して停止しない．}$$
$$f(M,M) = 0 \text{ のとき，} M \text{ は入力 } M \text{ に対して } 1 \text{ を出力して停止する．}$$

となる．ゆえに

$$f(M,M) = 1 \text{ のとき，} f(M,M) = 0.$$
$$f(M,M) = 0 \text{ のとき，} f(M,M) = 1$$

となって矛盾する．□

上記の証明は，いわゆる対角線論法の一種である．

3　算術的階層

計算可能性の概念を少し緩めた条件について考える．また，そうした条件をみたす集合どうしについて，計算困難性の度合いを比較する尺度を考える．本節ではアルファベットとして $\{0,1,B\}$ を用いる（B は空白記号）．以下では細かいことは気にせず，自然数全体の集合と $\{0,1\}^*$ を区別せずに論じる．

3.1　Recursively enumerable 集合（Computably enumerable 集合）

定義 **3.1**　1. プログラム（チューリング計算機）M と集合 $L \subseteq \{0,1\}^*$ が次の関係をみたすとき，M は L を決定する（判定する）という．すべての $x \in \{0,1\}^*$ に対し，

$$\begin{cases} x \in L \to M(x) = 1 \\ x \notin L \to M(x) = 0 \end{cases}$$

2. 以下が成り立つとき, M は L を半決定するという．すべての $x \in \{0,1\}^*$ に対し,

$$\begin{cases} x \in L \to M(x) = 1 \\ x \notin L \to M(x)\uparrow \text{（停止しない）} \end{cases}$$

ここででもう一つ，次のような条件を考えよう．

$(*)$ すべての $x \in \{0,1\}^*$ に対し，　$x \in L \leftrightarrow M(x) = 1$

上の条件 $(*)$ において $x \notin L$ の場合，M は停止しないかもしれないし，停止はするが 1 以外の出力をするかもしれない．与えられた言語 L に対し，以下の関係が成り立つ．

定義 5.1 (1) の条件をみたす M が存在 $\Rightarrow$ 定義 5.1 (2) の条件をみたす M が存在
$\Leftrightarrow$ $(*)$ をみたす M が存在

最初の $\Rightarrow$ と，後の（$\Leftrightarrow$ の）$\Leftarrow$ を示すには，1 以外を出力しようとしたら待ったをかけ，暴走するようにマシンを改造すればよい．さて，定義 5.1 の用語については様々な方言がある．例をあげよう（表 1）．

（注１）[渡辺 1992] の該当箇所ではどんな入力に対しても 1 または 0 を出力するプログラムだけを考えているので，この場合本書定義 5.1 (1) の条件と $(*)$ は同値になる．

「決定する」と「判定する」の違いは，単に「decide」の訳し方の違いに過ぎないと考えられる．

	定義 5.1 (1)	定義 5.1 (2)	($*$)
本稿	決定する（判定する）	半決定する	
Sipser [Si2008b, 164-165]	判定する（decide）		認識する（recognize）
[荻原 2006, p.14]	受理する（accept）		認識する（recognize）
[鹿島 2008, p.107]	決定する	半決定する	
[田中 1997, p.77]	受理する		
[渡辺 1992, 62-64]	認識する	半認識する	認識する（注１）
本稿旧版（2014）	決定する（認識する）	受理する（半認識する）	

表 1: 決定・判定・認識などについての方言

定理 3.2 集合 $L \subseteq \{0,1\}^*$ に対して，以下は同値．

1. L を半決定するプログラム M が存在する．
2. ある部分帰納的関数 $f : \{0,1\}^* \to \{0,1\}^*$ が存在して，f の定義域が L に等しい．
3. $L = \emptyset$ であるか，またはある全域的帰納的関数 $g : \{0,1\}^* \to \{0,1\}^*$ が存在して，g の像 $\{g(x) : x \in \{0,1\}^*\}$ が L に等しい．
4. ある帰納的述語 $R(-,-)$ があって，$L = \{x \in \{0,1\}^* : \exists w \in \{0,1\}^* \ \ R(x,w)\}$．
5. L は０型言語である（すなわち，ある句構造文法 G が存在して，G が L を生成する）．

証明の概略　L が有限集合のときは (1) から (5) までのすべてが成り立つと容易にわかる．以下，L が無限集合である場合のみ考える．

(1)⇔(2) の証：チューリング計算可能部分関数 ⇔ 部分帰納的関数 による．

(1)⇒(3) の証：M は (1) の条件をみたすとする．このとき，ビット列 x が L に属するための必要十分条件は以下で与えられる．

$\exists z \in \{0,1\}^* \, \exists s \in \mathbb{N}$ [z は M の計算過程（のコード），入力は x，出力は 1，ステップ数は s.]

ここで各組 (x,z,s) に対して，[ ] 内の条件はアルゴリズムによって有限時間で判定できる．そこで（$\mathbb{N}^2$ が可算無限集合であることの証明と同様にして）あらゆる組 (x,z,s) を $\{(x_i,z_i,s_i)\}_{i=0,1,2,\dots}$ と一列に並べるアルゴリズムを用いて，以下の操作によって $f(i)$（$i = 0,1,2,\dots$）を定める．

(x_i,z_i,s_i) が [ ] 内の条件を満たす最小の i を探し，それを i_0，$f(0) = x_{i_0}$ とする．

i_n と $f(n)$ が決まったとき，(x_i,z_i,s_i) が [ ] 内の条件を満たし，$i > i_n$ となる最小の i を探し，それを i_{n+1}，$f(n+1) = x_{i_{n+1}}$ とする．

(3)⇒(4) の証：L が全域的帰納的関数 g の像であるとする．このとき $L = \{z \in \{0,1\}^* : \exists x \in \{0,1\}^*\ \ g(x) = z\}$ と表せる．

(4)⇒(1) の証：ある帰納的述語 $R(-,-)$ があって，$L = \{x \in \{0,1\}^* : \exists w \in \{0,1\}^*\ \ R(x,w)\}$ となるとする．次のようなマシン M を考える．入力 x が与えられたとしよう．$R(x,w)$ がなりたつ w がみつかるまで，$w = 0,1,2,\cdots$ と w を一つずつ増やしていく．そのような w がみつかったとき 1 を出力して終了する．みつからなければいつまでも作業を続けていく．この M は L を半決定する．

(2)–(4) が同値であることのもう少しくわしい証明は [渡辺 1992, § 3.2] 参照．

0 型言語，句構造文法の定義は [田中 1997, § 2.4] 参照．(3) と (5) の同値性はそれほど難しくはない．本稿では略す．□

定義 3.3 (1)–(5) のいずれかが成り立つとき，L を *recursively enumerable set* という．別名： r.e. set，computably enumerable set，c.e. set，枚挙可能集合，帰納的可算集合，再帰的可算集合．

例 任意の帰納的集合は r.e. 集合である（帰納的 ⇔ 決定するプログラムあり．決定するプログラムを改造して半決定するプログラムを作れ）．

例 停止問題 $\mathrm{HALT} := \{\langle N, x\rangle : N$ は Tm（のコード）であり，入力 x に対して N は有限ステップで停止する．$\}$ は帰納的集合ではないが，r.e. 集合である．これが r.e. 集合であることを見るには，停止するための必要十分条件が以下で与えられることに注意せよ．

$$\exists y \in \{0,1\}^*\ \ T_1(N,x,y) = 0$$

定理 3.4 $L \subseteq \{0,1\}^*$ とその補集合がともに r.e. 集合であるとする．このとき L は帰納的集合である．

証明の概略　直観的にいうと，合格者名読み上げプログラムと不合格者名読み上げプログラムを用いて合否判定プログラムを作る．もう少し詳しくいうと次の通り．L および L の補集合を枚挙する全域的帰納的関数をそれぞれ f，g とする．入力 x が与えられたとき，$f(0), g(0), f(1), g(1), \cdots$ というように $n = 0,1,2,\cdots$ に対して順番に $f(n), g(n)$ を印字していき，$f(n) = x$ となる n がみつかったら 1 を出力して停止し，$g(n) = x$ となる n がみつかったら 0 を出力して停止するプログラムを考える．このプログラムは L を決定する．□

帰納的集合全体の族を $\mathcal{REC}$，r.e. 集合全体の族を $\mathcal{RE}$ と表そう．また，補集合が r.e. であるような集合全体の族を $\mathrm{co}\mathcal{RE}$ とする．このとき，$\mathcal{REC} = \mathcal{RE} \cap \mathrm{co}\mathcal{RE}$ 以下が成り立つ．また，$\mathcal{REC}$，$\mathcal{RE}$，$\mathrm{co}\mathcal{RE}$ は相異なる．

3.2 R.e. 完全集合

二つの集合の複雑さを比較する尺度を導入しよう．直観的にいうと，B をカンニングペーパーとして用いれば A を決定できるとき，A の複雑性は B の複雑性以下であると考える．より詳しくは以下の通り．

定義 3.5 A，B は $\{0,1\}^*$ の部分集合であるとする．

1. 次の条件をみたす関数 h を A から B への *many-one reduction* という．
   (a) h は $\{0,1\}^*$ から $\{0,1\}^*$ への全域的関数．
   (b) $\forall x \in \{0,1\}^* \ [x \in A \leftrightarrow h(x) \in B]$
   (c) h は計算可能（全域的帰納的関数）．
2. 上のような h が存在するとき，「A は B へ *many-one reducible* である」といい，$A \leq_{\mathrm{m}} B$ と書く．

Many-one reduction を多対一還元，m 還元ともいう．還元の代わりに帰着と訳すこともある．$\leq_{\mathrm{m}}$ は擬順序 (pseudo ordering) である．すなわち，反射律と推移律をみたす．ただし反対称律はみたさない．たとえば，偶数（の二進表記）全体の集合と奇数全体の集合は互いに many-one reducible である．

定義 3.6 集合 A が　$\forall L \in \mathcal{RE}\ \ L \leq_{\mathrm{m}} A$　かつ　$A \in \mathcal{RE}$　という条件をみたすとき，A を（$\leq_{\mathrm{m}}$ に関する）*r.e.-complete set*（*r.e.* 完全集合）という．

定理 3.7 停止問題 HALT は r.e.-complete である．

証明の概略　L を任意の recurisively enumerable set とする．定理 3.2 により，L を半決定する Tm M がある．そのような M をひとつ固定する．$\{0,1\}^*$ から $\{0,1\}^*$ への全域的関数 h を $h(x) = \langle M, x \rangle$ と定義する．すると h は L から HALT への many-one reduction となる．□

3.3　相対的計算可能性

オラクル チューリング マシン（OTm）は，通常の Tm にオラクルテープとクエリー状態という特別な状態を付け加えたものである．$N^{\sim}$ が OTm のとき，テープアルファベット上の語の集合 A を固定した場合のみ，クエリー状態直後の時点表示が定まる．

クエリー状態で，オラクルテープに書かれた語が A に属している場合：オラクルテープに書かれた語を 1（の 1 進コードすなわち 11）に書き換えた時点表示を，次のステップの時点表示とする．

クエリー状態で，オラクルテープに書かれた語が A に属していない場合：オラクルテープに書かれた語を 0（の 1 進コードすなわち 1）に書き換えた時点表示を，次のステップの時点表示とする．

集合 A をオラクルと呼ぶこともある．オラクル A を固定したとき「N^A が集合 L を決定（decide）する」という概念が自然に定義される．

ある OTm $N^{\sim}$ が存在して，N^A が L を決定する

とき，

L は recursive in A である

などという．

オラクル チューリング マシンの定義の詳細は文献によって微妙に異なる．[Sh2001, p.40] にややあいまいな形で述べられている定義を筆者（鈴木）が少し詳しくしたのが上記の定義である．[Pa1994, p.339] および [AB2009, p.73] では，クエリー状態の直後に OTm はテープに返答するのではなく，q_{yes}, q_{no} のどちらかの状態に遷移する．また，ここで述べたものとかなり違う計算モデルであっても，なんらかの外部情報を利用するものをオラクル チューリング マシンと呼ぶこともある．

3.4 算術的階層

自然数上の関係 $R(x_1,\dots,x_k,y_1,\dots,y_n)$ が算術的（arithmetical）とは，帰納的関係 $P(x_1,\dots,x_k,y_1,\dots,y_n)$ が存在して以下が成り立つことを言う．

$$R(x_1,\dots,x_k) \iff Q_1y_1\cdots Q_ny_nP(x_1,\dots,x_k,y_1,\dots,y_n),$$

ただし各 Q_i は $\forall$ か $\exists$ である．

$Q_1y_1\cdots Q_ny_n$ の部分を prefix といい，$P(x_1,\dots,x_k,y_1,\dots,y_n)$ を matrix という．（$n=0$ の場合についての一文がここにあったが，削除した．$n=0$ の場合は考えないことにする．）

Σ^0_1 型 prefix：$\exists y_1$	Π^0_1 型 prefix：$\forall y_1$
Σ^0_2 型 prefix：$\exists y_1\forall y_2$	Π^0_2 型 prefix：$\forall y_1\exists y_2$
Σ^0_3 型 prefix：$\exists y_1\forall y_2\exists y_3$	Π^0_3 型 prefix：$\forall y_1\exists y_2\forall y_3$
$\dots$	$\dots$
R が Σ^0_n 関係 $\iff_{def.}$	R が Π^0_n 関係 $\iff_{def.}$
R は prefix が Σ^0_n 型で	R は prefix が Π^0_n 型で
matrix が recursive	matrix が recursive
（なる関係と集合として等しい）	（なる関係と集合として等しい）

（$n=0$ の場合についての 3 行がここにあったが，削除した．）

R が Δ^0_n 関係 $\iff_{def.}$ R は Π^0_n 関係かつ Σ^0_n 関係．

命題 3.8

1. R が算術的関係 $\iff$ ある n があって R が Σ^0_n 関係または Π^0_n 関係
2. R が r.e. 関係 $\iff$ R が Σ^0_1 関係
3. co-r.e. $\iff$ R が Π^0_1
4. recursive $\iff$ Δ^0_1

停止問題の非可解性を用いて Δ^0_1(関係である k 項関係全体) $\subsetneq \Sigma^0_1$(同上)，$\Delta^0_1 \subsetneq \Pi^0_1$，$\Sigma^0_1 \neq \Pi^0_1$ がわかる．少し努力すると

$$\Delta^0_2 \iff \text{recusive in 停止問題}$$

もわかる．

参考文献

[AB2009] Arora, S., and Barak, B., *Computational complexity: A modern approach*. Cambridge (2009).（文献追加）

[En2001] Enderton, H. B., *A mathematical introduction to logic, 2nd ed.* Academic Press (2001).

[Pa1994] Papadimitriou, C., *Computational complexity*. Addison-Wesley (1994).

[Sh2001] Shoenfield, J. R., *Recursion theory*. A K Peters (2001).

[Si2008b] Sipser, M. 著, 太田ほか 訳, 「計算理論の基礎 [原著第２版] ２．計算可能性の理論」. 共立出版 (2008).

[荻原 2006] 荻原 光徳, 「複雑さの階層（アルゴリズム・サイエンスシリーズ—数理技法編)」. 共立出版 (2006).

[鹿島 2008] 鹿島 亮,「Ｃ言語による 計算の理論」. サイエンス社 (2008).

[田中 1997] 田中 尚夫, 「情報の数理 計算論理入門」. 裳華房 (1997).

[渡辺 1992] 渡辺治, 「計算可能性・計算の複雑さ入門」. 近代科学社 (1992).

鈴木登志雄「計算可能性理論 第 I 部 計算可能性」
2012 年 9 月 初版
2019 年 3 月 1 日 改訂版 2019.03.01
2020 年 4 月 1 日 改訂版 2020.04.01，12 月 13 日 revision a
2023 年 9 月 11 日 数学基礎論サマースクール版「計算論速習」，9 月 13 日 revison a