決定不能問題

Eureka GAP

2017 年 5 月 17 日

概要

この PDF のテーマは決定不能問題（undecidable problem）です．決定不能問題とはそれを解くための機械的なアルゴリズムが存在しない問題のことです．しかし，そもそもアルゴリズムが存在するということを数学的にどう定式化するのでしょうか？さらに，アルゴリズムが存在しないということはどう証明できるのでしょうか？プログラムの停止性問題をはじめとした決定不能問題の証明を目標として，再帰理論の基礎の基礎を紹介したいと思います．

0　諸注意

今回の発表において，**部分関数 (partial function)** とは自然数全体 $\mathbb{N} = \{0, 1, 2, \ldots\}$ の部分集合上で定義され，自然数の値をとるような写像とする．部分関数が**全域的（totally）**であるとは，すべての自然数において定義されることである．以後，全域的部分関数のことを単に**関数（function）**とよぶことにする[*1]．また，n 項関係 R とは $\mathbb{N}^n$ の部分集合を指し，$R(x_0, \ldots, x_{n-1})$ とは $(x_0, \ldots, x_{n-1}) \in R$ のことである．関係や集合はしばしばその特徴関数（characteristic function）：

$$\chi_R(x_0, \ldots, x_{n-1}) = \begin{cases} 0 & R(x_0, \ldots, x_{n-1}) \text{ のとき} \\ 1 & \neg R(x_0, \ldots, x_{n-1}) \text{ のとき} \end{cases}$$

と同一視される．また，記法上の注意として，変数の列 $x_0, \ldots, x_{n-1}$ を $\vec{x}$ と略す．そして，関数の値が定義されていることを $\downarrow$ で表す．定義されていないことを $\uparrow$ で表す．

1　再帰的関数

再帰理論または計算理論と現在よばれている分野は 1930 年代に誕生し，「アルゴリズムが存在する」という概念ないし「計算可能である」という概念を数学的に定式化する試みから始まった．今回の発表でも計算可能を定義することから始める．その前にこの章では再帰的部分関数というものを定義する．これは基本的な関数からいくつかの簡単な操作で得られる部分関数を指しており，再帰的であるならば「計算可能」であろうことは定義を見れば納得できるだろう．

[*1] 文献によって関数といった場合に全域的関数をさすのか，部分関数をさすのかは異なるため注意が必要である．

Definition 1.1. 部分関数の集合 Φ が**原始再帰的に閉じている（primitive recursively closed）**とは，次の条件 (1) から (3) を満たすことである．また，**再帰的に閉じている（recursvely closed）**とは，それに加えて条件 (4) も満たすことである．記号 $\simeq$ は両辺が定義されかつ値が等しいか，もしくはどちらも定義されていないことを表している．

(1) 初期関数 : Φ は，ゼロ関数 $\mathrm{Z}() = 0$，後続関数 $\mathrm{S}(x) = x+1$，射影関数 $\mathrm{P}^n_k(x_0, x_1, \ldots, x_{n-1}) = x_k$ を含む．

(2) 合成 : $G(y_0, y_1, \ldots, y_{n-1}), H_i(\vec{x})(i = 0, 1, \ldots, n-1)$ が Φ に含まれるとき，

$$F(\vec{x}) \simeq G(H_0(\vec{x}), H_1(\vec{x}), \ldots, H_{n-1}(\vec{x}))$$

も Φ に含まれる．

(3) 原始再帰 : $G(\vec{x}), H(\vec{x}, y, z)$ が Φ に含まれるとき，

$$F(\vec{x}, 0) \simeq G(\vec{x})$$
$$F(\vec{x}, y+1) \simeq H(\vec{x}, y, F(\vec{x}, y))$$

で定義した $F(\vec{x}, y)$ も Φ に含まれる．

(4) 最小化 : $G(\vec{x}, y)$ が Φ に含まれるとき，

$$F(\vec{x}) \simeq \mu y.G(\vec{x}, y)$$

も Φ に含まれる．ここで，右辺は

$$\mu y.G(\vec{x}, y) \simeq z \Leftrightarrow \forall y < z(G(\vec{x}, y)\downarrow \wedge G(\vec{x}, z) \neq 0) \wedge G(\vec{x}, y) \simeq 0$$

と定義される．

初期関数に合成，原始再帰を有限回適用して得られる（全域的）関数全体は原始再帰的に閉じている最小の集合である．また，初期関数に合成，原始再帰，最小化を有限回適用して得られる部分関数全体は再帰的に閉じている最小の集合である．

Definition 1.2. 原始再帰的に閉じている最小の集合に含まれる関数のことを**原始再帰的関数 (primitive recursive function)** とよぶ．再帰的に閉じている最小の集合に含まれる部分関数のことを**再帰的部分関数（recursive partial function）**とよぶ．全域的な再帰的部分関数は単に再帰的関数とよぶ．

普通我々が扱っている関数や関係はすべて原始再帰的である．

Example 1.3.

(1) 次の関数，関係は原始再帰的である．

$$x \dot{-} y (= \max\{x - y, 0\}),\ +,\ \cdot,\ x^y,\ x = y,\ x < y.$$

(2) 関係 R, S を原始再帰的とすると次のような関係も原始再帰的．

$$\neg R,\ R \vee S,\ R \wedge S,\ \exists y < x R(\vec{z}, y),\ \forall y < x R(\vec{z}, y).$$

Example 1.4. 原始再帰的でなく再帰的である関数の例：

$$A(0,y) = y+1$$
$$A(x+1,y) = A(x,1)$$
$$A(x+1,y+1) = A(x,A(x+1,y))$$

で定義される **Ackermann 関数** $A(x,y)$.

2 レジスター機械

次に，この章ではレジスター機械とよばれる計算モデルを導入する．これもまた「計算可能」であろう部分関数のクラスを定義する．

レジスター機械（register machine）とは，次のものから構成される．

(1) レジスター：自然数を一つだけ格納する．可算個存在し，レジスターやその値は $R_i (i \in \mathbb{N})$ で表される．いわばメモリー．
(2) カウンター：自然数を一つだけ格納する．一つだけ存在し，プログラムの実行にかかわる．カウンターやその値は C で表される．
(3) プログラム格納庫：プログラムを格納する．

レジスター機械におけるプログラムとは, 命令の有限列であり, それぞれの命令には順に $(0),(1),\ldots,(N-1)$ と番号が割り振られている．このときの N はプログラムの長さとよばれる．命令は次の 3 種類のみである．

(1) INCREASE R_i：レジスター R_i の値を 1 だけ増加させ，カウンターの値を 1 だけ増加させる．
(2) DECREASE R_i, m：レジスター R_i の値が正の場合，その値を 1 だけ減少させてカウンターの値を m とする．0 の場合，カウンターの値だけを 1 だけ増加させる．
(3) GOTO m：カウンターの値を m にする．

レジスター機械に長さ N のプログラム P と入力値 $\vec{x}$ が与えられたとき，次のようにプログラムは実行され，出力される．

(1) 入力値 $\vec{x}$ は順にレジスター $R_1, R_2, \ldots$ に格納される．ほかのレジスターの値は 0 とされる．
(2) カウンターの値は 0 とされる．
(3) プログラム格納庫にプログラム P が格納される．
(4) カウンターの値と同じプログラム番号の命令が実行される．
(5) もしカウンターの値が N になったならば，プログラムは停止し R_0 が出力値となる．

Definition 2.1. レジスター機械に $\vec{x}$ を入力して，$f(\vec{x})$ が出力されるようなプログラムが存在する場合，部分関数 f は**レジスター機械計算可能**であるという．

レジスター機械はいわばもっとも単純なコンピュータであり，ある部分関数がレジスター機械計算可能ならば，それを計算するような「アルゴリズムが存在する」と言えるだろう．また，レジスター機械計算可能ならば「計算可能」と言ってもよさそうである．さらに，次のことが言える．

Lemma 2.2. 再帰的部分関数はレジスター機械計算可能である．

Proof. 再帰的部分関数の構成に関する帰納法による．

(1) 初期関数：それぞれに対するプログラムを実際に作ってやればよい．
- ゼロ関数 Z：(0)GOTO 1.
- 後続関数 S：(0)GOTO 2 (1)INCREASE R_0 (2)DECREASE R_1,1 (3)INCREASE R_0.
- 射影関数 P_k^n：(0)GOTO 2 (1)INCREASE R_0 (2)DECREASE R_k,1.

(2) 合成：$H_0, H_1, \ldots, H_{n-1}$ のプログラムを実行しその値を用いて G のプログラムを実行すればよい．

(3) 原始再帰：G のプログラムを実行し，その値を用いて H のプログラムを実行する．直前に得られた計算結果を用いながら H のプログラムを繰り返せばよい．

(4) 最小化：順に $G(\vec{x},0), G(\vec{x},1), \ldots$ を計算していき，その値が 0 となったときの y を出力するようなプログラムで計算される．

以上より，再帰的部分関数には必ずそれを計算するプログラムがあり，レジスター機械計算可能である． □

この定理の逆も成り立つことは次の章で扱う．

3 コーディング

この章ではレジスター機械計算可能ならば再帰的であることを示すためにコーディングとよばれる手法を導入する．このコーディングというアイデアは Gödel によるものであり，不完全性定理の証明においても重要な役割を果たした．まず，自然数の列をひとつの自然数にコーディングすることを考える．

自然数列 $x_0, x_1, \ldots, x_{n-1}$ に対して，関数 $\langle\cdot\rangle$ を

$$\langle x_0, x_1, \ldots, x_{n-1}\rangle = p_1^{x_0+1} p_2^{x_1+1} \cdots p_{n-1}^{x_{n-1}+1}$$

と定義する．ただし，p_k は k 番目の素数を表す．また，次が成り立つように関数 $(\cdot)_i, \mathrm{lh}, *$ を定義する．これらはそれぞれ，コードの復号，列の長さ，列の連結を表す関数である．

$$(\langle x_0, x_1, \ldots, x_{n-1}\rangle)_i = x_i \quad (i < n)$$
$$\mathrm{lh}(\langle x_0, x_1, \ldots, x_{n-1}\rangle) = n$$
$$\langle x_0, x_1, \ldots, x_{n-1}\rangle * \langle y_0, y_1, \ldots, y_{m-1}\rangle = \langle x_0, x_1, \ldots, x_{n-1}, y_0, y_1, \ldots, y_{m-1}\rangle$$

$((x)_i)_j$ は $(x)_{i,j}$ と略記する．また，「x は列のコードである」という関係を $\mathrm{Seq}(x)$ とする．

この定義は素因数分解の一意性に基づいている．自然数の列をコーディングする場合は掛け算をすればよく，それをデコーディングしたい場合は素因数分解の計算をすればよい．これらの関数，関係を用いてプログラムをコーディングする．また，同様に計算過程をコーディングし，プログラムが計算する部分関数を明示することができる．具体的には次のようにする：

(1) 命令 I のコード $\ulcorner I \urcorner$ は次のように定める．

$$\ulcorner \text{INCREASE } R_i \urcorner = \langle 0, i\rangle$$
$$\ulcorner \text{DECREASE } R_i, m \urcorner = \langle 1, i, m\rangle$$
$$\ulcorner \text{GOTO } m \urcorner = \langle 2, m\rangle$$

「x は命令のコードである」という関係 Instruction(x) は次のように書ける．

$$x = \langle 0, (x)_1 \rangle \vee x = \langle 1, (x)_1, (x)_2 \rangle \vee x = \langle 2, (x)_1 \rangle$$

(2) N 行からなるプログラム P のコード $\ulcorner P \urcorner$ は命令のコードを $x_0, x_1, \ldots, x_{N-1}$ として，

$$\ulcorner P \urcorner = \langle x_0, x_1, \ldots, x_{N-1} \rangle$$

と定める．「x はプログラムのコードである」という関係 Program(x) は次のように書ける．

$$\mathrm{Seq}(x) \wedge \forall i < \mathrm{lh}(x)[\mathrm{Instruction}((x)_i) \wedge ((x)_{i,0} = 1 \rightarrow (x)_{i,2} \leq \mathrm{lh}(x)) \wedge ((x)_{i,0} = 2 \rightarrow x_{i,1} \leq \mathrm{lh}(x))]$$

(3) プログラム $e = \ulcorner P \urcorner$ に $x = \langle \vec{x} \rangle$ を入力したとき，時刻[*2]s におけるカウンター C の値を求める関数を Counter(e, x, s)，レジスター R_i の値を求める関数を Register(i, e, x, s) とする[*3]．また，

$$\overline{\mathrm{Register}}(e, x, s) = \langle \mathrm{Register}(0, e, x, s), \ldots, \mathrm{Register}(e + \mathrm{lh}(x), e, x, s) \rangle$$

とする．これは時刻 s における全てのレジスターの状態を表していることになる．なぜなら，コーディングの定義から，値が操作されるレジスター番号は，プログラムのコード e よりも小さく，計算に関係するレジスター番号は高々 e または入力の長さ $lh(x)$ となるからである．

(4)「時刻 s で計算が終了する」という関係 Halt(e, x, s) は

$$\forall m < s[\mathrm{Counter}(e, x, m) < lh(e)] \wedge \mathrm{Counter}(e, x, m) = \mathrm{lh}(e)$$

と書ける．

(5) 計算過程のコード y とは，プログラムの開始から終了までの時刻 s におけるレジスターの状態 $\overline{\mathrm{Register}}(e, x, s)$ の列をコーディングしたものとする．プログラムが止まらず計算不能な場合，y は定義されない．すると，「プログラム e に $\vec{x}$（長さ k）を入力したときの計算過程のコードが y である」ことを主張する k 変数関係 $T_k(e, \vec{x}, s)$ は

$$\mathrm{Program}(e) \wedge \mathrm{Seq}(y) \wedge \mathrm{Halt}(e, \langle \vec{x} \rangle, \mathrm{lh}(y) \dot{-} 1) \wedge \forall s < lh(y)[(y)_s = \overline{\mathrm{Register}}(e, \langle \vec{x} \rangle, s)]$$

と書ける．また，計算過程のコードから出力を求める関数 $U(y)$ は

$$U(y) = (y)_{\mathrm{lh}(r) \dot{-} 1, 0}$$

と書ける．

(6) 以上より，$e = \ulcorner P \urcorner$ でコーディングされるプログラム P が計算する n 変数部分関数 F_n^P は

$$F_n^P \simeq U(\mu y.T_n(e, \vec{x}, y))$$

と表すことができる．

[*2] ここでは，計算開始時刻を 0 とし，1 つ命令を終えるごとに時刻は 1 増加するものとする．

[*3] これらの関数を実際に定義するのはやや面倒であるが，これらの値が直前のレジスター機械の状態と実行するプログラムより一意に定まることから，再帰的に定義できる．

この議論により，次を証明することができる．

Lemma 3.1. レジスター機械計算可能な部分関数は再帰的である．

Proof. 関数 $\langle\cdot\rangle, (\,)_i, \mathrm{lh}, *$ と関係 $\mathrm{Seq}(x)$ を明示的な形で定義すれば，これらは原始再帰的であることがわかる．このことから，関係 $\mathrm{Instruction}(x), \mathrm{Program}(x), \mathrm{Counter}(e,x,s), \mathrm{Register}(i,e,x,s)$, $\mathrm{Halt}(e,x,s)$ はすべて原始再帰的であり，さらに，$T_k(e,\vec{x},s), U(y)$ もまた原始再帰的であることも分かる．よって，任意のレジスター機械計算可能な部分関数は，それを計算するプログラムのコード e によって $U(\mu y.T_n(e,\vec{x},y))$ と書かれる部分関数と同値なので，再帰的である． □

前節の結果と合わせると次の定理が示されたことになる．

Theorem 3.2. 部分関数（関係）がレジスター計算可能であることと，再帰的であることは同値である．

レジスター機械以外にも様々な計算モデルが考案されている．例えば，チューリングマシンや λ 計算などもそうである．しかし，結局のところそれらはすべて同等であり，それらの計算モデルにおいて計算可能な部分関数のクラスは再帰的部分関数のクラスと一致することが証明された．そこで，Church は次のような主張（定義）をした．

Definition 3.3. 計算可能部分関数とは，再帰的部分関数のことである．

これは Church のテーゼとよばれていて，この正当性は今や誰もが認めている．我々もこの主張を認める立場をとる．つまり，

$$\text{計算可能} \iff \text{（計算するための）アルゴリズムが存在} \iff \text{再帰的} \iff \text{レジスター機械計算可能}$$

というわけである．

4 基本的定理

計算可能部分関数についての基本的な定理 4 つを述べる．

Theorem 4.1. [正規形定理　Normal Form theorem] ある自然数 e と原始再帰的 1 変数関数 U と原始再帰的 n 変数関係 T_n が存在して，任意の n 変数計算可能部分関数 F に対して，

$$F(\vec{x}) \simeq U(\mu y.T_n(e,\vec{x},y))$$

が成り立つ．

Proof. 前節において示した通り，任意の計算可能関数に対してそれを計算する（レジスター機械の）プログラムが存在するので，そのコードを e としてやればよい． □

この定理で計算可能な f に対してとった e のことを f の**指標（index）**という．また，プログラム e が計算する n 変数部分関数を $\{e\}^n$ と書く．つまり，

$$\{e\}^n(x) \simeq U(\mu y.T_n(e,\vec{x},y))$$

である．$\{e\}^n$ や T_n の n はしばしば省かれる．ほとんど同じことの言い換えなので証明は省くが，次の定理も成り立つ．

Theorem 4.2. [枚挙定理　Enumeration theorem] $\{e\}^n$ 全体は n 変数計算可能部分関数の全体と一致する[*4]．また，$(e, \vec{x}) \mapsto \{e\}^n(\vec{x})$ という部分関数は計算可能である．このような部分関数は**万能関数 (universal function)** とよばれる．

ここでいう万能プログラムさえ構成すれば，もはや個々のプログラムは不要で，すべての部分関数の計算を実行することができる．このような意味で「万能」なのである．

さらに，次の定理も重要である．

Theorem 4.3. [S-m-n 定理　S-m-n theorem] 各 $m, n \geq 1$ について $1+m$ 変数計算可能関数 S_n^m が存在して，任意の $e, \vec{y} = y_1, \ldots, y_m, \vec{x} = x_1, \ldots, x_n$ について

$$\{S_n^m(e, \vec{y})\}(\vec{x}) \simeq \{e\}(\vec{y}, \vec{x})$$

が成り立つ．

Proof. S_n^m は，$e, \vec{y}$ が与えられたとき，e がプログラム P の指標ならば次のようなプログラムの指標を返す全域的計算可能関数とすればよい：$\vec{x}$ が入力されたとき，その前に $\vec{y}$ を挿入し，P を実行する．　□

実はこの証明における S_n^m としては原始再帰的関数で単射なものをとることができる．今回は S_n^m が計算可能であることだけを用いるのでその証明は省略する．

Theorem 4.4. [再帰定理 Recursive theorem] $F(\vec{x}, y)$ を計算可能部分関数とすると，ある e が存在して，

$$\{e\}(\vec{x}) \simeq F(\vec{x}, e)$$

が成り立つ．

Proof. S-m-n 定理より，

$$\{S_n^1(y, y)\}(\vec{x}) \simeq \{y\}(\vec{x}, y)$$

となる計算可能関数 S_n^1 が存在する．$F(\vec{x}, S_n^1(y, y))$ は再帰的部分関数なので，枚挙定理よりその指標 d がとれて，$e = S_n^1(d, d)$ とすれば，

$$\{S_n^1(d, d)\}(\vec{x}) \simeq \{d\}(\vec{x}, d) \simeq F(\vec{x}, S_n^1(d, d))$$

よりこれが存在を示したい e である．　□

最後に，半計算可能性を定義する．これは関数ではなく関係に対して定義されるものである．

[*4] このことを $\{e\}^n$ は計算可能部分関数を枚挙する，という．

Definition 4.5. 計算可能部分関数 $\{e\}$ の定義域を $W_e = \{\vec{x} : \{e\}(\vec{x})\downarrow\}$ と書くことにして，

$$\exists e \forall \vec{x}(R(\vec{x}) \leftrightarrow \vec{x} \in W_e)$$

が成り立つ関係 R を**半計算可能（semicomputable）**とよぶ[*5]．また，このときの e を半計算可能な関係の指標とよぶ．

R が計算可能ということはその特徴関数が計算可能ということなので，$R(\vec{x})$ が成り立つならば 0 を返し，$R(\vec{x})$ が成り立たないならば 1 を返すようなアルゴリズムが存在する．一方，関係 R が半計算可能であるとき存在が言えるアルゴリズムは，$R(\vec{x})$ が成り立たない場合には停止しない．

Theorem 4.6. 計算可能ならば半計算可能である．

このことは，次の定理から直ちに導かれることである．

Theorem 4.7. 関係 R が半計算可能であることの必要十分条件は，ある計算可能関係 S により，

$$\forall \vec{x}[R(\vec{x}) \leftrightarrow \exists y S(\vec{x}, y)]$$

となることである．

Proof. 必要性：R の指標を e として，正規形定理における $T_n(e, \vec{x}, y)$ を S とすればよい．十分性：$\mu y.S(\vec{x}, y)$ の指標を e とすれば，これは R の指標である． □

5 決定不能問題

さて，いよいよ本題である決定不能問題について述べる．決定不能問題とは，それを解くための機械的なアルゴリズムが存在しない問題のことであった．ここでは大きく分けて 3 種類の決定不能問題をあげる．すなわち，停止性問題（と Rice の定理），語の問題，公理系の決定性問題である．

Theorem 5.1. [停止性問題　Halting problem] 与えられた入力に対してプログラムは停止するかという問題は決定不能である．すなわち，

$$\{(y, x) : \{y\}(x)\downarrow\}$$

は計算不能．

Proof.

$$K = \{x : \{x\}(x)\downarrow\}$$

とおく．K の補集合が半計算可能としてその指標 e をとる．このとき，$e \in W_e$ と $e \notin W_e$ が導かれ矛盾．補集合が半計算可能，特に計算可能でないので K も計算可能でない． □

この証明は**対角線論法（diagonal method）**の一例となっている．

[*5] 再帰的枚挙可能（recursively enumerable, RE）とも言う．

Corollary 5.2.

(1) あるプログラム e が存在して，それに対する停止性問題 $\{x : \{e\}(x)\downarrow\}$ は計算不能．よって，部分関数が全域的であるかを判定する問題もまた決定不能．
(2) 入力を 0 に限定しても停止性問題は決定不能，すなわち $\{e : \{e\}(0)\downarrow\}$ は計算不能．

Proof. (1) 枚挙定理により $\{e\}(x) \simeq \{x\}(x)$ となる e をとればよい．後半は前半から明らか．(2) S-m-n 定理より $\{S(x,x)\}(0) \simeq \{x\}(x,0)$ となる計算可能関数 S が存在し，$\{e : \{e\}(0)\downarrow\}$ が計算可能なら $\{x : \{x\}(x,0)\downarrow\}$ も計算可能となるが，これは先の定理と同様に計算不能であると示すことができる． □

$K = \{x : \{x\}(x)\downarrow\}$ は計算可能でなく，半計算可能な集合の例となっている．半計算可能であることは系 (1) の証明における計算可能部分関数 $\{e\}$ の定義域となっていることから分かる．また，$\mathbb{N} \setminus K$ は半計算可能でない集合の例となっている．

停止性問題のような指標の集合は計算可能でない場合が多い．次の定理を示すことで多くの決定不能問題，計算可能でない集合が得られる．

Theorem 5.3. [Rice の定理] 計算可能部分関数の集合 $\mathcal{F}$ についてその指標全体の集合 $A = \{e : \{e\} \in \mathcal{F}\}$ が計算可能となるのは，$\mathcal{F}$ が空または計算可能部分関数全体である場合に限る．

Proof. $\mathcal{F} \neq \emptyset, \mathbb{N}$ とする．A が計算可能であると仮定する．このとき，$F \in \mathcal{F}, G \notin \mathcal{F}$ となる計算可能部分関数 F, G が存在し，

$$H(x,y) \simeq \begin{cases} G(x) & y \in A \text{ のとき} \\ F(x) & y \notin A \text{ のとき} \end{cases}$$

と定義される H は計算可能である．再帰定理より $\{e\}(x) \simeq H(e,x)$ となる e がとれるが，$e \in A$ と $e \notin A$ が同時に導かれて矛盾．よって A は計算可能でない． □

Example 5.4. Rice の定理からわかる計算可能でない集合の例.

(1) $A_0 = \{e : \{e\}(0)\downarrow\}$（この例は **Corollary 5.2** でも示した.）
(2) $A_1 = \{e : \{e\}$ は空関数 $\}$
(3) $A_2 = \{e : \{e\}$ は全域的 $\}$

次に語の問題（Word problem）について述べる．その前に少し定義をしておく．

Definition 5.5. **半 Thue システム（semi-Thue system）**とは，記号の有限集合 Σ と生成規則の集合 $\mathcal{P}$ の組 $(\Sigma, \mathcal{P})$ をさす．Σ に含まれる記号の有限列（空列を除く）を **Σ-語（Σ-word）**とよぶが，普通は単に語とよばれる．生成規則は $\alpha \to \beta$（α, β は語）と表現される．語 X に生成規則 $\alpha \to \beta$ を適用するとは，X 内に現れる α のうち 1 つを β にして新たな語を生成することである．さらに，有限回の生成規則の適用により，語 X から語 Y が生成されたとき，$X \Rightarrow_T Y$ と書き，Y は X から導出可能であるという．また，任意の語 X に対して，$X \Rightarrow_T X$ とする[*6].

Theorem 5.6. [半 Thue システムにおける語の問題] 任意の半 Thue システム $T = (\Sigma, \mathcal{P})$ と任意の語 X, Y が与えられたとき，Y が X から導出可能であるかどうかは決定不能である．

Proof. 特に $(\Sigma, \mathcal{P})$ と Y を固定しても語の問題が決定不能であることをみる．方針としては，レジスター機械のプログラム P を半 Thue システムにおいて再現し，Y が X から導出可能であることと P が停止することが同値になるように $(\Sigma, \mathcal{P})$ と Y をうまく設定する．

あるレジスター機械のプログラム P をとる．P は N 行のプログラムからなり，計算にかかわるレジスターは M 番未満とする．このとき，Σ には 1, b, b_i $(i \leq M)$, c_i $(i \leq N)$, d_i, e_i $(i < N)$ が含まれているものとする．

レジスターの状態に対応する語は

$$bc_n b_1 1^{r_0} b_2 1^{r_1} \cdots 1^{r_{M-1}} b_M$$

である．ここで r_i はレジスター R_i の値を表し，1 が x 回続くことを 1^x と書いている．c_n はカウンターの値が n であることを示す．b, b_i は単なる区切りである．また，

$$I(x) = bc_0 b_0 b_1 1^x b_2 \cdots b_M$$

を初期状態語とよぶ．そして，P の各命令に対して生成規則を導入する．

(1) (n)INCREASE R_i という命令に対応し，

$$c_n b_j \to b_j c_n \ (j < i),\ c_n 1 \to 1 c_n,\ c_n b_i \to d_n b_i 1,\ 1 d_n \to d_n 1,\ b_j d_n \to d_n b_j (j < i),\ b d_n \to b c_n$$

という生成規則を導入する．

(2) (n)DECREASE R_i, m という命令の $R_i \neq 0$ の場合に対応し，

$$c_n b_j \to b_j c_n \ (j < i),\ c_n 1 \to 1 c_n,\ c_n b_i 1 \to d_n 1,\ 1 d_n \to d_n 1,\ b_j d_n \to d_n b_j (j < i),\ b d_n \to b c_m$$

という生成規則と，$R_i = 0$ の場合に対応する

$$c_n b_i b_{i+1} \to e_n b_i b_{i+1},\ 1 e_n \to e_n 1,\ b e_n \to b c_{n+1}$$

という生成規則を導入する．

(3) (n)GOTO m という命令に対応し，

$$b c_n \to b c_m$$

という生成規則を導入する．

(4) プログラムの終了に対応し，

$$c_N b_i \to c_N \ (0 \leq i \leq M),\ c_N 1 \to c_N$$

という生成規則を導入する．

*6 これは本質的な制約ではなく，後の議論の都合でつけたものである．

このように定めた半 Thue システムにおいて，c_i，d_i，e_i がちょうど 1 つだけ含まれる語を状態推移的であるとよぶことにする．明らかに bc_N 以外の状態推移的な語に対して適用できる生成規則はただ 1 つであり，それを適用してできる語もまた状態推移的である．bc_N に適用できる生成規則はない．また，初期状態語 $I(x)$ は状態推移的である．

このとき，$I(x) \Rightarrow_T bc_N$ であることとプログラム P が入力 x のもとで停止することは同値となる．なぜなら，生成規則はプログラム P の計算を模倣して適用されていくので，初期状態 $I(x)$ に生成規則を適用していくと状態推移的な語の列が一意的に得られ，P が停止するときに限り bc_N に到達するからである．

ここで，**Corollary 5.2**(1) において存在がいえた停止性問題が決定不能な P に対応する半 Thue システムを考える．半 Thue システムにおける語の問題が決定可能ならば，プログラムの停止性問題も決定可能となるが，これは矛盾である． □

この決定不能性の証明における重要な点は，そのシステムの内部でレジスター機械の動きを模倣することができるということにある．つまり，そのシステムはレジスター機械と同様の計算能力を持つ計算モデルとみなせる．特に，万能関数を計算するレジスター機械と同様の計算能力をもつ場合，その計算モデルは**チューリング完全（Turing complete）**であるという．半 Thue システムはまさにチューリング完全である．そして，チューリング完全な計算モデルにおいては必ずプログラムの停止性問題に対応する決定不能問題がある．

半 Thue システムの語の問題はかなり限定的な問題に見えてしまうかもしれないが，実は少し工夫するだけでより興味深い決定不能問題を得ることができる．最後にこのことを見ていくことにする．

半 Thue システム，またはその生成規則の集合 $\mathcal{P}$ が対称的であるとは，生成規則 $\alpha \to \beta$ が $\mathcal{P}$ に含まれるならば $\beta \to \alpha$ も $\mathcal{P}$ に含まれるということである．先の証明で構成した半 Thue システム $T = (\Sigma, \mathcal{P})$ について考える．$\mathcal{P}$ を含む対称的な最小の生成規則の集合を $\mathcal{P}'$ とし，半 Thue システム $(\Sigma, \mathcal{P}')$ を T' とおく．しかし，実はこの生成規則の拡大は決定不能性には影響を与えない．

Lemma 5.7. 上のように定義した半 Thue システム T, T' に対し，

$$I(x) \Rightarrow_{T'} bc_N \iff I(x) \Rightarrow_T bc_N$$

である．

Proof. 十分であることは明らかなので，必要性を示す．$I(x)$ から bc_N まで T' における生成規則を順に適用して得た状態推移的語の列を $X_0, \ldots, X_{k-1}$（X_0 は $I(x)$，X_{k-1} は bc_N）とおく．ここで長さ k は最小のものをとる．$0 < i < k$ において，$X_{i-1} \Rightarrow_T X_i$ ならばよい．そうでなければ，$X_{i-1} \Rightarrow_T X_i$ とならない最大の i をとる．明らかに $i \neq k-1$．このとき，$X_i \Rightarrow_T X_{i-1}$ かつ $X_i \Rightarrow_T X_{i+1}$ が成り立つが，状態推移的語に生成規則を適用して得られる語は一意なので X_{i-1} と X_{i+1} は一致して，これは語の列の長さの最小性に反するので矛盾． □

この結果から，次は明らかである．

Lemma 5.8. 対称的な半 Thue システムの語の問題は決定不能である．

半群とは，結合律の 2 項演算が定義されている集合をさすが，語全体の集合は語の並置を積とみなすことで半群となる．これは Σ 上の**自由半群（free semigroup）**とよばれる．そして，対称的な半 Thue システムの生成規則 $\alpha \to \beta$ は $\alpha = \beta$ という条件を与えていることになり，$X \Rightarrow_{T'} Y$ はそれらの条件から $X = Y$ とい

う帰結が得られたことを意味する．よって，対称的な半 Thue 系における議論は自由半群における議論として解釈できて，次が成り立つ．

Theorem 5.9. [半群の語の問題] 有限集合 Σ が与えられたとして，Σ 上の自由半群を S とする．さらに，$\alpha = \beta(\alpha, \beta \in S)$ という形の有限個の条件と $X, Y \in S$ が与えられたときに，$X = Y$ かどうかを判定する問題は決定不能．

実は 3 つ目の決定不能問題である公理系の決定問題（Decision problem）はこの定理からの帰結である．ここで，公理系が決定不能であるとは，論理式が公理系から導かれるのかを判定するアルゴリズムが存在しないということである．ここの箇所だけは基本的な論理学の知識を仮定する．

Theorem 5.10. 半群の公理系は決定不能．

Proof. **Theorem 5.9** を証明するために構成した記号の集合 Σ とその自由半群 S，$\alpha = \beta$ の形の有限個の条件の集合 $\mathcal{D}$ をとる．Σ に属する記号に対して定数記号を半群の言語 L に追加した言語を L' とおく．Σ に属する記号と L' の定数記号は同一視する．Σ 上の語は L' 上の閉項とみなせる．また，半群の公理（系）: $\forall x, y, z((x \cdot y) \cdot z = x \cdot (y \cdot z))$ を SG とおき，SG のみを含む L'-理論を T' とする．ここで，$\mathcal{D}$ に属する条件をすべて連言で結んだ論理式を D とする．このとき，$T' \vdash D \to X = Y$ （X, Y は語）の判定問題は決定不能である．また，$\Sigma = \{a_0, \ldots, a_n\}$ とすれば，証明中に現れる定数をすべて変数に置き換えることにより任意の L'-式 $\phi[a_0, \ldots, a_{n-1}]$ に対して，$T' \vdash \phi[a_0, \ldots, a_{n-1}] \Leftrightarrow SG \vdash \forall x_0 \ldots x_{n-1} \phi[x_0, \ldots, x_{n-1}]$ が成り立つ．ゆえに，

$$T' \vdash D \to X = Y \Leftrightarrow SG \vdash \forall x_1 \ldots x_n (D \to X = Y)[x_1, \ldots, x_n]$$

が導ける．しかし，左辺は決定不能であるから右辺もまた決定不能である． □

Theorem 5.11. 言語 L は 2 変数関数記号を少なくとも一つ含むとする．このとき L-論理式が論理的に帰結されるかどうかは決定不能．

Proof. $SG \vdash \phi \Leftrightarrow \vdash SG \to \phi$ であるが，半群の公理系は決定不能なので，右辺は決定不能である．ここで半群における積を L に含まれる 2 変数関数記号とみなせばよい． □

参考文献

[1] 新井敏康 著『数学基礎論』岩波書店，2011.

[2] A.J. Kfoury, R.N. Moll, M.A. Arbib 共著，甘利俊一，金谷健一，川端勉 共訳『プログラミングによる計算可能性理論（A Programing Approach to Computability）』サイエンス社，1987.

[3] 田中一之 編・著，鹿島亮，角田法也，菊池誠 著『数学基礎論講義』日本評論社，1997.

[4] 篠田寿一 著『帰納的関数と述語』河合出版，1997.

[5] J.R. Shoenfield, "Recursion Theory," Springer, 1993.

[6] H. Rogers, "Theory of Recursive Functions and Effective Computability," The MIT Press, 1987.