計算可能性理論（序）　改訂版 2020.04.01 revision a

東京都立大学 理 数理　　鈴木 登志雄

目 次

1　序

1.1　計算可能性，計算複雑性を学ぶ意義

計算機に何ができるか？どれだけの計算資源が必要なのか？計算機を理想化・抽象化した概念，すなわち**計算モデル**を導入することにより，上記の問題を特定のハードウェアに依存しない形で論じていこう．

Bruce et al.[BDKT2003] は，計算機科学の学生が数学を学ぶ意義としてとくに，効率的なアルゴリズムの作成をあげている．例として，オリンピック期間中のある街において，タクシーの効率的な配車計画を行うプログラムを作ることを考える．彼らはプログラムへの要求を 3 種類導入し，一見するとささいな違いだがそうではないことを説く．要求 A は強欲アルゴリズムによって $n \log n$ オーダーの時間で処理できる．要求 B は動的計画法によって $n \log n$ オーダーで処理できる．要求 C は NP 困難であり，近似アルゴリズムを使うなどの妥協が必要になる．

このようなアルゴリズムの設計と解析のためにも，計算可能性・計算複雑性の基本知識は有用なのである．

1.2　計算モデルの例：レジスター マシン

計算モデルの例として，まずレジスター マシン [Co2004, §2.3]（以下，略称 Rm）をみる．Rm はレジスター $R_1, R_2, \ldots$ をもち，これらは自然数 $r_1, r_2, \ldots$ を格納する．Rm は 4 種の基本命令をもつ．行番号付きの基本命令を有限個並べたものを Rm **プログラム**という．

Rm の基本命令

r_n = 0	/* R_n の内容を 0 にする． */
r_n = r_n + 1	/* R_n の内容をインクリメントする（1 増やす）． */
r_n = r_m	/* R_n に R_m の内容を上書きコピーする． */
`if(` r_n `==` r_m `){ goto k; }`	/* もし R_n と R_m の内容が等しいなら k 行目にジャンプする． */

例 1.1 Rm プログラムの例.

```
1:  if( r2 == r3 ){ goto 5; }
2:   r1 = r1 + 1
3:   r3 = r3 + 1
4:  if( r1 == r1 ){ goto 1; }
```

$r_1 = 5, r_2 = 2, r_n = 0 (n \geq 3)$ として例 1.1 のプログラムを実行してみよう．すなわち，まず 1 行目を実行する．特定の行へのジャンプを指定されたときはそれに従い，そうでなければ次の行を実行する．次に実行すべき行が存在しないときは，その時点でプログラムが停止するものと約束する．停止するまで作業を続ける．

実行すべき行，r_1, r_2, r_3 を時系列に並べると次の通りとなる．

$(1,5,2,0)$, $(2,5,2,0)$, $(3,6,2,0)$, $(4,6,2,1)$, $(1,6,2,1)$, $(2,6,2,1)$, $(3,7,2,1)$, $(4,7,2,2)$, $(1,7,2,2)$, $(5,7,2,2)$ 停止

例題 1.2 自然数 a, b が与えられたとき $r_1 = a, r_2 = b, r_n = 0 (n \geq 3)$ として例 1.1 のプログラムを実行するとき，実行すべき行，r_1, r_2, r_3 を時系列に並べるとどうなるか考察せよ.

解 $b = 0$ の場合　$(1,a,b,0)$, $(5,a,b,0)$ 停止

$b > 0$ の場合　$(1,a,b,0)$, $(2,a,b,0)$, $(3,a+1,b,0)$, $(4,a+1,b,1)$, $(1,a+1,b,1)$, $\cdots$, $(1,a+b,b,b)$, $(5,a+b,b,b)$ 停止 □

定義 1.1　1. Rm プログラム P が関数 $f : \mathbb{N}^k \to \mathbb{N}$ を**計算する**とは，任意の $(x_1, \ldots, x_k) \in \mathbb{N}^k$ に対して以下が成り立つことをいう．$r_1 = x_1, \ldots, r_k = x_k, r_n = 0 \quad (n > k)$ としてプログラム P を実行すると，基本命令を有限回実行した後にマシンは停止し，しかもそのとき $r_1 = f(x_1, \ldots, x_k)$ となっている.

2. 関数 $f : \mathbb{N}^k \to \mathbb{N}$ が *Rm* **で計算可能である**とは，f を計算する Rm プログラムが存在することをいう.

例題 1.3 自然数の加算 $f(x,y) = x + y$ が Rm で計算可能であることを示せ.

解　例題 1.2 により，例 1.1 の Rm プログラムは $f(x,y) = x + y$ を計算する．よって自然数の加算は Rm で計算可能である. □

興味深いことに，計算モデルの定義として Rm 以外のものを採用しても，計算可能性の概念は変わらないことが多い．第 I 部ではチューリング計算機という計算モデルについて考察し，帰納的関数とチューリング計算可能関数の等価性を示す．その証明をよくみれば，Rm による計算可能性もやはりこれらと同値であることがわかる．さらに，計算不可能な関数の例を紹介し，計算不可能な集合について計算困難性の度合いを論じる．第 II 部では，計算可能な関数に対して，計算に必要な資源，すなわち計算複雑性を考察する．とくに多項式時間計算量クラス P に属する集合の具体例を紹介すること，および NP 完全性・NP 困難性という概念を紹介することに重点を置く.

1.3　有限オートマトン

準備運動として，序の残りの部分で有限オートマトンと正規言語について概観する．これらの部分を読みとばしても第 I 部以降を読むのに論理的に支障はない．有限オートマトンは，記憶能力に制約を与えられた計算モデルの一種である．正式な定義に先立って，おおまかな説明をしておこう．レジスターマシンで行番号というものを考えた．たとえば例 1.1 では次のようになっていた.

入力：計算開始の時点で 1 番レジスターと 2 番レジスターに格納されている自然数
行番号の集合：$\{1, 2, 3, 4\}$
最初に処理する行番号：1
停止の合図：行番号が 5 以上になったとき
出力：停止後に 1 番レジスターに格納されている自然数

有限オートマトンは行番号と似た内部状態という概念をもつ。しかし，レジスターに相当する部分が貧弱，というよりほとんどないのである．たとえば，これは一つの例であるが，次のようになっている．

入力：なんらかの文字列（左端から 1 文字ずつ読んでいく）
内部状態の集合：$\{q_0, q_1, q_2, q_3, q_H\}$
初期状態：q_0
停止の合図：入力を読み切ったとき
出力：停止後の状態が q_H なら Yes，そうでなければ No

有限オートマトンは，入力を左端から 1 文字ずつ読み込み，1 文字読むごとに内部状態を変えることができる．これは，レジスターマシンが行を一つ処理するたびに「次に処理すべき行」を変えていくのと似ている．では，定義を見ていこう．空でない有限集合（を記号の集合と考えたもの）を**アルファベット**という．

定義 1.2 アルファベット Σ が与えられたとする．

1. ある自然数 n に対し，Σ の要素を重複を許して n 個並べたものを Σ 上の *word*（**語**），あるいは *string*（有限列）という．n をその語の**長さ**といい，語 α の長さを $|\alpha|$ で表す．長さ 0 の語を**空語**（あるいは**空列**）といって λ で表す[1]．例：$\alpha = 1101$ のとき $|\alpha| = 4$.

2. 語 α と β をこの順に並べて一つの語としたものを α と β の**連接**といい，$\alpha\beta$ で表す．例：$\alpha = hot$，$\beta = tea$ のとき $\alpha\beta = hottea$.

3. Σ 上の長さ n の語全体の集合を Σ^n で表す．また，Σ 上の語全体の集合を Σ^* で表す．Σ^* の部分集合を Σ 上の *language*（**言語**）という．

「状態 q で入力文字 a を読み込んだとき，次の状態を q' とする」ということを表すために，「$\delta(q, a) = q'$」となる関数 δ を導入したい．詳しくは，以下のように定義する．

定義 1.3 $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ が**有限オートマトン**であるとは，以下が成り立つことをいう．

- Q は空でない有限集合．Q の要素を**内部状態**，あるいは単に状態という．
- Σ はアルファベットで，$Q \cap \Sigma = \emptyset$．$\Sigma$ の要素を**入力記号**という．
- δ は $Q \times \Sigma$ から Q への関数．**状態遷移関数**という．
- q_0 は Q の要素．**初期状態**という．
- F は Q の部分集合．F の要素を**受理状態**という．

[1] 文献によっては ε で表す．

「状態 q で入力文字列 $a_1a_2a_3$ を左端から 1 文字ずつ読み込んだとき，状態は q' になっている」ということも関数で表したい．これには，状態遷移関数を自然に拡張すれば十分である．$q_1 = \delta(q, a_1)$，$q_2 = \delta(q_1, a_2)$ とすれば，上記の主張は「$\delta(q_2, a_3) = q'$」で表される．そこで関数 δ を拡張し，$\delta(\delta(\delta(q, a_1), a_2), a_3)$ を $\delta(q, a_1a_2a_3)$ と書くことにすれば，上記の主張は「$\delta(q, a_1a_2a_3) = q'$」と表せる．一般に $\delta(q, \lambda) = q$ とし，$\delta(q, a_1a_2 \cdots a_n)$ は上記と同様に定める．このようにして，関数 $\delta : Q \times \Sigma \to Q$ を $Q \times \Sigma^*$ から Q への関数に拡張する．

定義 1.4 $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ は有限オートマトンとする．

1. 語 w が $\delta(q_0, w) \in F$ をみたすとき（つまり初期状態から始めて w を読み切ると受理状態に至るとき），w を M によって**受理される語**という．
2. M によって受理される語全体の集合を M によって**受理される言語**という．

例 1.4 $Q = \{q_0, q_1, q_2\}$，$\Sigma = \{0, 1\}$，$F = \{q_2\}$ とし，δ を表 1 で与える．

	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_0

表 1: 状態遷移表

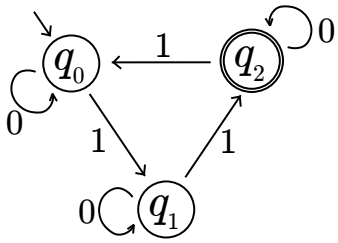

図 1: 状態遷移図

この有限オートマトン $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ が受理する言語は以下の集合である．

$$\{w \in \{0,1\}^* : w \text{ における } 1 \text{ の出現回数が } (3 \text{ の倍数}) + 2\}$$

1.4 正規言語

定義 1.5 $G = \langle V_N, V_T, S, P \rangle$ が**句構造文法**（略して「文法」）であるとは以下が成り立つことをいう．

- V_N はアルファベット．V_N の元を *nonterminal symbol*（非終端記号）あるいは**変数**という．
- V_T もアルファベットで，$V_N \cap V_T = \emptyset$．V_T の元を *terminal symbol*（終端記号）という．
- S は V_N の要素で，**開始記号**という．
- P は，$u \to v$ という形の式の有限集合．ただし u, v は $(V_N \cup V_T)$ 上の語であり，u は少なくとも一つ nonterminal symbol をもつ．P の元を**生成規則**という．

定義 1.6 $G = \langle V_N, V_T, S, P \rangle$ は文法とする．

1. $(V_N \cup V_T)$ 上の語で，$\alpha_1 u \alpha_2$ の形をしたものを考える．生成規則 $u \to v$ が P の中に入っているとき，「$\alpha_1 u \alpha_2$ は G **によって** $\alpha_1 v \alpha_2$ **に移る（移行する）**」という．$\alpha_1 u \alpha_2 \underset{G}{\to} \alpha_1 v \alpha_2$ と書く．

2. α から出発して G による移行を有限回（0 回も可）して β を得るとき,「β は G によって α から導出される」といい，$\alpha \xrightarrow[G]{*} \beta$ と書く．

3. とくに $S \xrightarrow[G]{*} \beta$，かつ $\beta \in V_T^*$ のとき（すなわち，終端記号だけの語 β が開始記号から導出されているとき），β は G から生成されるという．

4. G から生成された終端語全体の集合を G から生成された言語という．

例 1.5 $V_N = \{(\text{文}), (\text{主語}), (\text{述語})\}$，開始記号は (文)，$V_T = \{(\text{白鳥は}), (\text{フナは}), (\text{飛ぶ}), (\text{泳ぐ})\}$，
$P = \{(\text{文})\to(\text{主語})(\text{述語}),\ (\text{主語})\to(\text{白鳥は}),\ (\text{主語})\to(\text{フナは}),\ (\text{白鳥は})(\text{述語})\to(\text{白鳥は})(\text{飛ぶ}),$
(白鳥は)(述語)→(白鳥は)(泳ぐ), (フナは)(述語)→(フナは)(泳ぐ)} とする．
このとき,「(白鳥は)(飛ぶ)」を次のように生成できる．
(文) → (主語)(述語) → (白鳥は)(述語) → (白鳥は)(飛ぶ)
同様にして「(白鳥は)(泳ぐ)」および「(フナは)(泳ぐ)」を生成できる．一方,「(フナは)(飛ぶ)」は生成できない．

例 1.6 1. 空集合 $\emptyset$ を生成する文法を考えよう．非終端記号は開始記号 S のみ，終端記号は a のみ，生成規則は $S \to aS$ のみとする．すると終端記号だけの語を生成することはできないので，この文法が生成する言語は $\emptyset$ である．

2. $\{a, b, c\}$ を生成する文法を考えよう．たとえば，非終端記号は開始記号 S のみ，終端記号は a，b，c，生成規則は $S \to a$，$S \to b$，$S \to c$ とすればよい．同様にして， 一般に有限集合 $\{a_1, \ldots, a_n\}$ を生成する文法を作ることができる．

以下，同じ文字 a が n 回連続して並んだ文字列を表すのに a^n という表記を用いることがある．たとえば $aaabbcccc$ は $a^3b^2c^4$ と表せる．

例 1.7 $\{a^n b : n = 0, 1, 2, \ldots\}$ を生成する文法を考えよう．たとえば，次のようにすればよい．$S \to aS$，$S \to b$.

例 1.8 $\{a^n b^n : n = 1, 2, \ldots\}$ を生成する文法を考えよう．たとえば，次のようにすればよい．$S \to aSb$，$S \to ab$.

定義 1.7 (1) 文法 $G = \langle V_N, V_T, S, P\rangle$ が *regular grammar*（**正規文法**）であるとは，どの生成規則も以下二つのパターンのどちらかの形をしていることをいう．ただし大文字は非終端記号，小文字は終端記号を表すものとする．
$A \to aB$
$A \to a$
左辺が開始記号 S であるときのみ，例外として
$S \to \lambda$
を許す．ただしこの例外的な規則があるときは，他の生成規則の右辺に開始記号が現れないものとする．
(2) 正規文法から生成された言語を *regular language*（**正規言語**）という．

1.5 有限オートマトンと正規文法の等価性

以下では M がオートマトンのとき M が受理する言語を $T(M)$ で表し，G が文法のとき G が生成する言語を $L(G)$ で表す．

定理 1.8 $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ は有限オートマトンとする．このとき，ある正規文法 $G = \langle V_N, V_T, S, P\rangle$ が存在して $T(M) = L(G)$ となる．

証明の概要：話を簡単にするため，M が空語を受理しない場合のみ考えるが，空語を受理する場合への一般化はそれほど難しくない．$V_N = Q$，$V_T = \Sigma$，$S = q_0$ とする．さらに $P_1 = \{q \to ap : \delta(q,a) = p\}$，$P_2 = \{q \to a : \delta(q,a) \in F\}$，$P = P_1 \cup P_2$ として，文法 $G = \langle V_N, V_T, S, P\rangle$ を定義する．G はたしかに正規文法の条件をみたしている．このとき以下が成り立つ．

$$
\begin{aligned}
& a_1a_2\cdots a_n \in L(G) \\
\Leftrightarrow\ & q_0 \overset{*}{\underset{G}{\longrightarrow}} a_1a_2\cdots a_n \\
\Leftrightarrow\ & q_0 \underset{G}{\longrightarrow} a_1q_1 \underset{G}{\longrightarrow} \cdots \underset{G}{\longrightarrow} a_1\cdots a_{n-2}q_{n-2} \underset{G}{\longrightarrow} a_1\cdots a_{n-2}a_{n-1}q_{n-1} \underset{G}{\longrightarrow} a_1\cdots a_n \\
\Leftrightarrow\ & q_0 \underset{G}{\longrightarrow} a_1q_1,\ \cdots,q_{n-2} \underset{G}{\longrightarrow} a_{n-1}q_{n-1} \text{ かつ } q_{n-1} \underset{G}{\longrightarrow} a_n \\
\Leftrightarrow\ & \delta(q_0,a_1) = q_1,\ \cdots,\delta(q_{n-2},a_{n-1}) = q_{n-1},\ \text{かつ } \delta(q_{n-1},a_n) \in F \\
\Leftrightarrow\ & \delta(q_0,a_1\cdots a_n) \in F \\
\Leftrightarrow\ & a_1a_2\cdots a_n \in T(M)
\end{aligned}
$$

よって $L(G) = T(M)$ □

定理 1.9 $G = \langle V_N, V_T, S, P\rangle$ は正規文法とする．このとき，ある有限オートマトン $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ が存在して $T(M) = L(G)$ となる．

証明の概要：正規文法 G が与えられたとする．定理 1.8 と大体同じように話を進める．ただし文法の場合，たとえば $S \to aS$，$S \to a$ のように，左辺が同じなのに右辺が異なる生成規則があるかもしれない．一方，有限オートマトンは，(q,a) が決まれば関数値 $\delta(q,a)$ は一通りに決まるので，このままではうまくいかない．そこで有限オートマトンの定義を少し改造し，各 $\delta(q,a)$ が Q の要素ではなく Q の部分集合になるようにする．たとえば $\delta(q_0,a) = \{q_1,q_2\}$ の場合，状態 q_0 で入力文字 a を読むと次の状態は q_1 か q_2 のどちらかをランダムに選ぶと解釈する．すると入力が与えられたとき，計算経路（計算の運命）は一通りには決まらない．語 w が受理されるとは，受理状態に至る計算経路が少なくとも一つ存在することと定める．このようにして非決定性有限オートマトンを定義する．すると定理 1.8 の証明と同様にして，$T(N) = L(G)$ となる非決定性有限オートマトン $N = \langle Q', \Sigma, \delta', q_0, F'\rangle$ の存在を示せる．

次に，$T(N) = T(M)$ となる有限オートマトン M の存在を示したい．まず Q' の空でない部分集合全体からなる集合族を Q とする．$\delta(\{r_1,\ldots,r_m\},a) = \delta'(r_1,a) \cup \cdots \cup \delta'(r_m,a)$ によって δ を定める．$\{r_1,\ldots,r_m\}$ が M の受理状態であるとは $r_1,\ldots,r_m$ の少なくとも一つが F' に属することと定め，M の受理状態全体を F とおく．このようにして有限オートマトン $M = \langle Q, \Sigma, \delta, \{q_0\}, F\rangle$ を定める．少し努力すると $T(N) = T(M)$ を示せる． □

1.6　正規言語でない言語の例

補題 1.10（pigeonhole principle） n は自然数，集合 A の要素の個数は $n+1$（以上）で，集合 B の要素の個数は n（以下）とする．このとき A から B への単射は存在しない．

この補題は証明抜きに用いる．大学初年級向けの証明は，たとえば [鈴木 2016] の第 1 章の章末問題 1.9 とその解答（p.139）を参照．

定理 1.11（pumping lemma，ポンプ補題）L は正規言語とする．このとき，ある正の整数 r（L のポンプ定数とよばれる）が存在して，L の元で長さ r 以上の語 w は（もしこのような w があればそのどれも）以下のような分解 $w = xyz$ をもつ．

- $|y| \geq 1$ かつ $|xy| \leq r$
- すべての $i \geq 0$ に対して，$xy^iz \in L$

証明の概要：L を受理する有限オートマトン $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ を一つ固定する．$r = |Q|$ とおく．$w = a_1 \cdots a_m \in L$，$|w| = m \geq r$ とする．w を上記のように分解できることを示そう．$f(0) = q_0$ とする．また，$1 \leq i \leq r$ となる i に対して，M が w の左から i 文字目まで読み込んだ直後の状態を $f(i)$ とする．pigeonhole principle により，$f : \{0, 1, \ldots, r\} \to Q$ は単射ではない．したがって $0 \leq j < k \leq r$，$f(j) = f(k)$ となる j, k が存在する．

$x = a_1 \cdots a_j$，$y = a_{j+1} \cdots a_k$ と定める．残りの部分（空語かもしれない）を z とする．すなわち，$w = xyz$．任意の $i \geq 0$ に対し，M が xy^i を読み込んだ直後の状態は $f(k)$ であるから，M が xy^iz を読み込んだ直後の状態は，$w = xyz$ を読み込んだ直後の状態と同じ受理状態である．よって $xy^iz \in L$ となる．　□

例 1.9 言語 $L = \{a^nb^n : n \geq 1\}$ は正規言語ではない．

証明の概要：背理法の仮定として L が正規言語だとすると，ポンプ補題により L のポンプ定数 r が存在する．$w = a^rb^r \in L$ であるから，w の分解 $w = xyz$ で，ポンプ補題の性質をみたすものがある．

場合 1 :y が文字 a と b を両方含む場合．このとき $y = a^mb^\ell$ の形になる（ただし $m, \ell \geq 1$）．$xy^2z = xa^mb^\ell a^mb^\ell z$ となるが，これは「前半が a のみで後半が b のみ」という形をしていないから L に属さない．ポンプ補題に矛盾する．

場合 2 :それ以外の場合。y は 1 種類の文字だけからなる．たとえば $y = a^m$（$m \geq 1$）の場合，$xy^2z = a^{r+m}b^r$ となり，これは a の出現回数と b の出現回数が異なるから L に属さない．ポンプ補題に矛盾する．

以上により背理法によって，L が正規言語でないことが示された．　□

執筆に際し，レジスターマシンについては [Co2004] を，有限オートマトンについては [田中 1997] を参考にした．

参考文献

[BDKT2003]　Bruce, K.B., Drysdale, R. L. S., Kelemen, C., Tucker, A., “Why math?” *Communications of the ACM*, 46, pp. 40–44 (2003).

[Co2004]　Cooper, S. B., *Computability theory.* Chapman and Hall/CRC (2004).

[鈴木 2016]　　鈴木 登志雄, 「例題で学ぶ集合と論理」. 森北出版 (2016).

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

鈴木登志雄「計算可能性理論（序）」
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