

平成 13 年 10 月 29 日作成

平成 15 年 9 月 16 日改訂

# コンピュータのための情報数理

早稲田大学工学部  
経営システム工学科

平澤 茂一



# 序

コンピュータは数多くの科学・工学・技術の結集物である。しかもその裾野は広く、したがってあらゆる自然科学は勿論、人文科学・社会科学が関連している。とりわけコンピュータとその周辺の問題について、研究・開発・設計したり、機能や性能を記述・解析・合成・評価する際、その道具となる理論が必要である。純粋な数学ももちろんコンピュータに不可欠であるが、ここでは直接情報工学・情報技術に近いものを指している。これが本書で紹介する情報数理解(情報科学・情報数学もほぼ同義語)である。ちょうど電気回路を設計・解析する際、基礎として電磁気学・回路理論(過渡現象論・交流理論を含む)が必要であるように、コンピュータを設計・評価するのに基礎として情報数理解が必要なのである。しかし、電気工学や機械工学などの歴史のある工学とは異なり、いずれもせいぜいここ 50 年、永くても 100 年くらいの比較的新しい学問で、十分体系化されているとは言えない。しかも、コンピュータや情報を取り扱うのにオールマイティな理論があるわけではない。複数の理論から構成された情報数理解にはそれぞれの得意とする守備範囲があり、目的・対象・立場・論点・性質などに応じ使い分けなければならない。

ここでとり上げなかった重要なものに、学習理論・数値解析・ファジー集合論・シミュレーション技法・予測理論・最適化理論・信頼性理論・トラフィック理論などの他、集合・代数系・グラフ理論など離散数学といわれるものがある。

なお、情報科学はアメリカでは**計算機科学**(computer science)、フランスでは**情報学**(informatics)、ロシアでは**サイバネティクス**(cybernetics)などとよばれる。

本書は情報数理解とよばれる、情報を取り扱うための基礎となる理論をコンパ

クトで紹介した学部用のテキストである。半年の授業の前半でこれを用いて紹介し、後半でこの中の一つを選んで詳しく解説することを想定している。勿論、毎回一章ごとを紹介し、演習などで補足する方法もある。

なお、本書は内山明彦・平澤茂一共著、「理工系のための計算機工学」(昭晃堂、平成2年)を大幅に改定する際、第1章、1.2節 計算機と情報科学 を思い切って別冊としたものである。

平成13年10月

著者記

# 目 次

<b>1</b>	<b>ブール代数とスイッチング理論</b>	<b>1</b>
1.1	ブール代数の公理系 . . . . .	1
1.2	ブール関数 . . . . .	2
1.3	標準形 . . . . .	3
1.4	双対原理 . . . . .	4
<b>2</b>	<b>数理論理</b>	<b>7</b>
2.1	シンタックスとセマンティックス . . . . .	7
2.2	命題論理 . . . . .	9
2.3	述語論理 . . . . .	11
<b>3</b>	<b>オートマトンと言語</b>	<b>15</b>
3.1	有限状態オートマトン . . . . .	15
3.2	形式言語 . . . . .	18
<b>4</b>	<b>チューリングマシンと計算論</b>	<b>23</b>
4.1	チューリングマシン . . . . .	23
4.2	計算論 . . . . .	25
<b>5</b>	<b>アルゴリズムと計算量</b>	<b>31</b>
5.1	アルゴリズムの定義 . . . . .	31
5.2	アルゴリズムの構成法 . . . . .	32
5.3	計算量の評価 . . . . .	36
5.4	問題の複雑さのクラス . . . . .	38

<b>6</b>	<b>プログラム理論</b>	<b>43</b>
6.1	プログラムの意味論 . . . . .	44
6.2	プログラムの証明論 . . . . .	44
<b>7</b>	<b>情報理論</b>	<b>47</b>
7.1	情報量の定義 . . . . .	47
7.2	情報源符号化 . . . . .	50
7.3	通信路符号化 . . . . .	53
<b>8</b>	<b>符号理論</b>	<b>55</b>
8.1	誤り訂正・検出のしくみ . . . . .	56
8.2	重要な符号 . . . . .	58
<b>9</b>	<b>暗号理論</b>	<b>63</b>
9.1	暗号系のモデルと暗号強度 . . . . .	63
9.2	秘密鍵暗号系 . . . . .	65
9.3	公開鍵暗号系 . . . . .	66
9.4	ゼロ (零) 知識証明 . . . . .	68

# 1

## ブール代数とスイッチング理論

ブール代数は1854年、G.Booleによって記号論理演算の場として導入された美しい数学の一分野である。現在では、もっと一般に束の概念に含まれる。ブール代数が論理設計に応用できることを指摘したのは1938年、C.E.Shannonである。スイッチング理論はリレー・真空管・トランジスタ・ダイオードなどスイッチ素子を用いたデジタル回路の論理設計を扱うもので、ブール代数を用いて展開される。ここでは、ブール代数について述べる。

なお、ブール代数は集合代数、数理論理学における命題論理(2.2節参照)の代数系と同型である。

### 1.1 ブール代数の公理系

集合 $\mathcal{B}(\neq \phi)$ において、 $\mathcal{B}$ 上の2つの2項演算 $+$ 、 $\cdot$ と1つの単項演算 $\bar{\phantom{x}}$ が定義されているとする。次の公理が成立するとき、 $\mathcal{B}$ をブール代数(Boolean algebra)といい、 $\langle \mathcal{B} : +, \cdot, \bar{\phantom{x}}, 0, 1 \rangle$ と書く。ここで、 $\phi$ は空集合を示す。

[定義 1.1] (ブール代数の公理 (I)) 代数系が次の性質を満足するとき、 $\mathcal{B}$ をブール代数という。 $\forall x, y, z \in \mathcal{B}$  に対し

$$(1)(\text{交換則}) \quad x + y = y + x, \quad x \cdot y = y \cdot x \quad (1.1)$$

$$(2)(\text{結合則}) \quad x + (y + z) = (x + y) + z, \quad x \cdot (y \cdot z) = (x \cdot y) \cdot z \quad (1.2)$$

$$(3)(\text{吸収則}) \quad x + (x \cdot y) = x, \quad x \cdot (x + y) = x \quad (1.3)$$

$$(4)(\text{分配則}) \quad x + (y \cdot z) = (x + y) \cdot (x + z), \quad x \cdot (y + z) = (x \cdot y) + (x \cdot z) \quad (1.4)$$

$$(5)(\text{補元}) \quad \exists \bar{x} \in \mathcal{B} : x + \bar{x} = 1, \quad \exists \bar{x} \in \mathcal{B} : x \cdot \bar{x} = 0 \quad (1.5)$$

$$(6)(\text{べき等則}) \quad x + x = x, \quad x \cdot x = x \quad (1.6)$$

$$(7)(\text{恒等元})^1 \quad \exists 0 \in \mathcal{B} : x + 0 = x, \quad \exists 1 \in \mathcal{B} : x \cdot 1 = x \quad (1.7)$$

が成り立つ。 □

<sup>1</sup> $1 = 0$ となるブール代数も考えられるが、自明なので $1 \neq 0$ としておく。

このように通常、ブール代数は束 (lattice) の公理 ((1),(2),(3)) に分配則 (4) と補元 (5) の条件を加えたもので、分配束かつ可補束で定義される (**ブール束**ともいう)。

ここで、これらの条件は互いに独立ではない。明らかに、(6),(7) はなくてもよい。また、例えば (1),(4),(5),(7) から (2) や (3) を導くことができる。

しかし、代数的構造を見やすくし演算を容易にするために、冗長な条件が加えられることが多い。**公理**から導くことができる結果を**定理**という。ブール代数では実用的な立場から公理・定理を区別せず、通常論理設計のために有用な条件式を**ブール代数の公式**として示す事が多い。

このように、公理系はいくつも存在する。次に演算則の独立性を確かめた最小の公理系を示す。

**[定義 1.2] (Huntington のブール代数の公理 (II))** 代数系が次の性質を満足するとき、 $\mathcal{B}$  をブール代数という。  $\forall x, y, z \in \mathcal{B}$  に対し

$$(1)(\text{恒等元}) \exists 0 \in \mathcal{B} : x + 0 = x, \quad \exists 1 \in \mathcal{B} : x \cdot 1 = x \quad (1.8)$$

$$(2)(\text{交換則}) x + y = y + x, \quad x \cdot y = y \cdot x \quad (1.9)$$

$$(3)(\text{分配則}) x + (y \cdot z) = (x + y) \cdot (x + z), \quad x \cdot (y + z) = (x \cdot y) + (x \cdot z) \quad (1.10)$$

$$(4)(\text{補元}) \exists \bar{x} \in \mathcal{B} : x + \bar{x} = 1, \quad \exists \bar{x} \in \mathcal{B} : x \cdot \bar{x} = 0 \quad (1.11)$$

が成り立つ。 □

数学的には最小の公理系を求めることは重要なことであるが、一般に唯一とは限らないし、これを求めるのは容易でないことが多い。

論理設計でしばしば用いられる**ドモルガン (de Morgan) の定理**ももちろん公理から証明されるべき結果なのである (演習問題 [1.1] 参照)。

## 1.2 ブール関数

$\mathcal{B} = \{0, 1\}$  とおく。このとき、 $\mathcal{B}_0 = \langle \mathcal{B} : +, \cdot, \bar{\phantom{x}}, 0, 1 \rangle$  と書くことにすると  $\mathcal{B}_0$  は最も簡単なブール代数のモデルである<sup>2</sup>。

$\mathcal{B}_0$  の  $n$  次の直積  $\mathcal{B}_0^n$  から  $\mathcal{B}_0$  への写像を  $n$  変数**ブール関数** ( $n$  変数論理関数) といい、 $\mathcal{B} = \{0, 1\}$  を定義域とする変数を**ブール変数** (論理変数) という。  $n$  変

<sup>2</sup> $\|\mathcal{B}\| = m$  とすると  $m = 2^\nu$  ( $\nu$  : 正整数) のときにしかブール代数のモデルが存在しないことが知られている。ここで、 $\|A\|$  は集合  $A$  の要素数を示す。



数ブール関数の集合を  $\mathcal{B}_n$  と書くと

$$\mathcal{B}_n = \{f | f : \mathcal{B}_0^n \rightarrow \mathcal{B}_0\} \quad (n = 1, 2, \dots) \quad (1.12)$$

$$\|\mathcal{B}_n\| = 2^{2^n} \quad (1.13)$$

である<sup>3</sup>。すなわち、 $n$  個のブール変数を  $x_1, x_2, \dots, x_n$  とするとき、 $(x_1, x_2, \dots, x_n) \in \mathcal{B}_0^n$  を  $f(x_1, x_2, \dots, x_n) \in \mathcal{B}_0$  とする関数  $f(\cdot)$  がブール関数である。また、 $m = 2^n$  個の元  $(x_1, x_2, \dots, x_n)$  の  $f(\cdot)$  の写像による対応表が**真理値表**である。一般に問題が与えられると、 $f_i(\cdot) (i = 1, 2, \dots, 2^m)$  のどれかが唯一きまる。

以上により、任意のブール関数は論理演算の組  $\{+, \cdot, \neg\}$  で展開できる (具体的には式 (1.16) 参照) ことが明らかとなった。このような論理演算の組を**万能演算系**という。 $\{+, \cdot, \neg\}$  は**万能演算系**であるが最小ではない。一方、 $\{+, \neg\}, \{\cdot, \neg\}$  は**最小万能演算系**である。

### 1.3 標準形

$x$  以外の変数  $x_1, x_2, \dots, x_n$  を含むブール形式<sup>4</sup>  $f(x)$  は

$$f(x) = f(0)\bar{x} + f(1)x \quad (1.14)$$

を満たす<sup>5</sup>。一般に

$$f(x_1, x_2, \dots, x_n) = f(0, x_2, x_3, \dots, x_n)\bar{x}_1 + f(1, x_2, x_3, \dots, x_n)x_1 \quad (1.15)$$

が成り立つ。これを**Shannon 展開**という。この操作を  $x_2, x_3, \dots, x_n$  についても続けると次の定理が得られる。

<sup>3</sup> $n = 2$  のとき、 $\|\mathcal{B}_2\| = 16$ 。また、 $\mathcal{B}_n$  を  $\langle \mathcal{B}_n : +, \cdot, \neg, 0, 1 \rangle$  と表せば、 $\mathcal{B}_n$  は要素数が  $2^{2^n}$  のブール代数である。

<sup>4</sup> $\mathcal{B} = \{0, 1\}$  上のブール変数と演算記号  $+, \cdot, \neg$  を用いてブール関数を表現したもの。

<sup>5</sup>演算記号  $+, \cdot, \neg$  の個数を  $k$  とするとき、 $k = 0$  を示した後、 $k \geq 1$  とし帰納法を用いて証明する。

[定理 1.1] (主加法標準形 (principal disjunctive normal form)) 任意の  $n$  変数ブール関数  $f(x_1, x_2, \dots, x_n)$  は次式の形で表現される.

$$f(x_1, x_2, \dots, x_n) = \sum_{(a_1, a_2, \dots, a_n) \in \mathcal{B}_0^n} f(a_1, a_2, \dots, a_n) x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \quad (1.16)$$

$$x_i^{a_i} = \begin{cases} x_i, & a_i = 1; \\ \bar{x}_i, & a_i = 0 \end{cases} \quad (1.17)$$

ここで,  $\sum$  はすべての  $(a_1, a_2, \dots, a_n) \in \mathcal{B}_0^n$  の加法 (論理和)  $+$  を表す. □

式 (1.16), (1.17) にドモルガンの定理 (式 (q.1.1.1), (q.1.1.2)) を適用すれば容易に次の定理が得られる.

[定理 1.2] (主乗法標準形 (principal conjunctive normal form)) 任意の  $n$  変数ブール関数  $f(x_1, x_2, \dots, x_n)$  は次式の形で表現される.

$$f(x_1, x_2, \dots, x_n) = \prod_{(a_1, a_2, \dots, a_n) \in \mathcal{B}_0^n} \left[ f(a_1, a_2, \dots, a_n) + \sum x_i^{\bar{a}_i} \right] \quad (1.18)$$

ここで,  $\prod$  はすべての  $(a_1, a_2, \dots, a_n) \in \mathcal{B}_0^n$  の乗法 (論理積)  $\cdot$  を表す. □

#### 1.4 双対原理

ブール代数の公理 (I) の (1) ~ (7) において, 各条件の 2 つの条件式 (式 (1.1) ~ 式 (1.7) の左右の 2 つの式) は, 一方の  $+$  と  $\cdot$ , 0 と 1 を入れかえると他方が得られる. 公理 (II) も同様である. このように, ある命題  $P$  の  $+$  と  $\cdot$ , 0 と 1 を入れかえて得られる命題  $P^*$  を  $P$  の**双対** (dual) という. ブール代数の公理の双対を作ると再び公理になる. これをブール代数の**双対原理** (duality principle)

という<sup>6</sup>. したがって, ある命題  $P$  を証明すればその双対となる命題  $P^*$  は証明する必要がない.

もう少し正確に言えば,  $\langle \mathcal{B} : +, \cdot, \bar{\phantom{x}}, 0, 1 \rangle$  で成り立つ命題は  $\langle \mathcal{B} : \cdot, +, \bar{\phantom{x}}, 1, 0 \rangle$  でも成り立つ. すなわち,  $f(x_1, x_2, \dots, x_n), g(x_1, x_2, \dots, x_n)$  をブール形式とすると,  $f(\cdot) = g(\cdot)$  ならば,  $f^*(\cdot) = g^*(\cdot)$  も成り立つ. ここで  $f^*(\cdot)$  を  $f(\cdot)$  の**双対関数**という.  $f^*(\cdot)$  は  $f(\cdot)$  の  $+$  と  $\cdot$ ,  $0$  と  $1$  を入れかえても作ることができるが

$$f^*(x_1, x_2, \dots, x_n) = \overline{f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)} \quad (1.19)$$

としても求められる (演習問題 [1.2] 参照).

[例 1.1] (双対関数の例)

(1)  $f(x_1, x_2) = x_1\bar{x}_2 + \bar{x}_1x_2$  のとき

$$f^*(x_1, x_2) = \overline{\bar{x}_1x_2 + x_1\bar{x}_2} = (x_1 + \bar{x}_2)(\bar{x}_1 + x_2)$$

(2)  $f(x_1, x_2) = f(0, 0)\bar{x}_1\bar{x}_2 + f(1, 0)x_1\bar{x}_2 + f(0, 1)\bar{x}_1x_2 + f(1, 1)x_1x_2$  のとき

$$\begin{aligned} f^*(x_1, x_2) &= \overline{f(0, 0)x_1x_2 + f(1, 0)\bar{x}_1x_2 + f(0, 1)x_1\bar{x}_2 + f(1, 1)\bar{x}_1\bar{x}_2} \\ &= \overline{(f(0, 0) + \bar{x}_1 + \bar{x}_2)(f(1, 0) + x_1 + \bar{x}_2)(f(0, 1) + \bar{x}_1 + x_2)(f(1, 1) + x_1 + x_2)} \end{aligned}$$

が成り立つ. □

双対関数の定義から

$$\overline{f^*(x_1, x_2, \dots, x_n)} = f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n) \quad (1.20)$$

が成り立つ. さらに,  $f^*(\cdot) = f(\cdot)$  のとき**自己双対関数**といい

$$\overline{f(x_1, x_2, \dots, x_n)} = f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n) \quad (1.21)$$

が成り立つ.

[例 1.2] (自己双対関数の例)

$f(x_1, x_2, x_3) = \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3$  のとき,  $f^*(x_1, x_2, x_3) = \overline{x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3} = (\bar{x}_1 + x_2 + x_3)(x_1 + \bar{x}_2 + x_3)(x_1 + x_2 + \bar{x}_3)(x_1 + x_2 + x_3) = \{(x_1 + x_2)(x_1 + \bar{x}_2) + x_3\}(x_1 + x_2) = x_1x_2 + x_2x_3 + x_1x_3 = f(x_1, x_2, x_3)$  となり,  $f(\cdot)$  は自己双対関数である. □

---

<sup>6</sup>束でも双対性が成り立つ.

**演習問題**

[1.1] ブール代数の公理系 (II) を用いて、**ドモルガンの定理**

$$\overline{x + y} = \bar{x} \cdot \bar{y} \quad (\text{q.1.1.1})$$

$$\overline{x \cdot y} = \bar{x} + \bar{y} \quad (\text{q.1.1.2})$$

を証明せよ.

[1.2] ブール関数  $f(x_1, x_2, \dots, x_n)$  の  $+$  と  $\cdot$ ,  $0$  と  $1$  を入れかえて得られた双対形  $f^*(x_1, x_2, \dots, x_n)$  は

$$f^*(x_1, x_2, \dots, x_n) = \overline{f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)} \quad (\text{q.1.2.1})$$

として得られたブール形式に等しいことを示せ.

[1.3] 最小の万能演算系の例を二つ以上示せ.

**参考文献**

1. 入江盛一, 数理論理学入門, 培風館, 1973年.
2. 赤堀也, 伊藤正夫, 後藤英一, 広瀬健編, 情報処理のための数学, 共立出版, 昭和50年.
3. 上坂吉則, 情報数学の基礎, 培風館, 昭和52年.
4. 上坂吉則, 船田哲男, 木俣昇, 情報科学の基礎, 培風館, 昭和54年.
5. 小野厚夫, 川口正昭, 情報科学概論, 培風館, 1983年.
6. 中川圭介, 計算機の論理設計, 近代科学社, 1984年.
7. 高松吉郎, 田代嘉宏, 情報の基礎数学, 培風館, 1988年.
8. 山田輝彦, 論理回路理論, 森北出版, 1990年.
9. 小倉久和, 高濱徹行, 情報の論理数学入門, 近代科学社, 1991年.
10. 道脇義正, 情報科学のための数学入門, 東京図書, 1991年.
11. 小野寛晰, 情報科学における論理, 日本評論社, 1994年.
12. 笹尾勤, 論理設計, 近代科学社, 1995年.
13. 田中尚夫, 計算論理学入門, 裳華房, 1997年.

## 2 数理論理

**論理** (logic, mathematical logic) とは、一般に言語機能をもった**形式的体系** (formal system) である。言語機能とは論理的記述のための機能とこれによって記述された**論理式** (well formed formula : wff) の正しさを証明する機能である。論理は判断機能を抽象化したものとみなせるから、“ある世界”を抽象的に**解釈** (interpretation) した結果と考えることができる。そのため、**知識情報処理**と深いかわり合いをもっている。

もともと、論理学はギリシャ時代アリストテレス (Aristotle) によって考えられた学問である。思考の法則を自然言語によって表現するのは正確さの点で十分ではない。そこで、論理学に記号的方法を適用し、事実の集合 (論理式集合) から**推論**という思考過程を数学的に取り扱えるようにしたものが**数理論理** (学) である。すなわち、思考過程はあらかじめ定められた規則 (公理と推論規則) によって形式的に論理式を変形する有限回の操作とみなせる。その結果、推論は計算と類似した概念となる。

J.McCarthy は早くから “世界”に関する情報をコンピュータにもたせるために論理の研究を行ってきた。ここで、解釈 (あるいはモデル) は自然で実際に即したように与えられる。したがって、解釈の仕方により結果は一通りではない。たとえば、第1章のスイッチング理論は、スイッチングを論理と定義した2.2に述べる命題論理とみなせる。以上の結果、個々の論理は形式的体系  $S$  と、論理式の解釈  $m$  によって規定される。

なお、論理式は論理の構成要素記号を文法に従って並べたもの、解釈は形式的体系の記号に対象とする世界の要素を対応させたものである。この結果、意味を与えていると考えられる。特に、文法を**シンタックス** (syntax)、意味を**セマンティックス** (semantics) という。

### 2.1 シンタックスとセマンティックス

#### (1) シンタックス

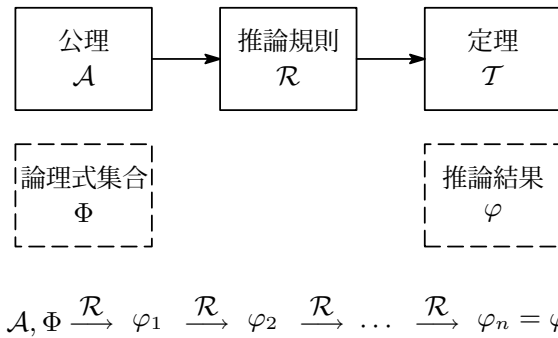
形式的体系  $S$  は論理式集合  $\Phi = \{\varphi_1, \varphi_2, \dots, \varphi_n\}$ 、**公理** (axiom)  $\mathcal{A}$ 、**推論規則** (inference rule)  $\mathcal{R}$  からなる (図 2.1)。 $\Phi$  と任意の論理式  $\varphi$  が  $\mathcal{R}$  の関係にあるとき、 $\varphi$  は  $\mathcal{R}$  によって  $\Phi$  から**直接導出可能**という。

[定義 2.1] (導出可能)  $S$  における  $\Phi \vee \varphi$  に対し,  $\varphi$  が  $\Phi$  から導出可能 (deducible) のとき  $\Phi \vdash \varphi$  と書く.  $\Phi \vdash \varphi$  とは論理式の列  $\{\varphi_1, \varphi_2, \dots, \varphi_n\}$  が存在し,  $\varphi = \varphi_n$  かつすべての  $i$  に対し, 次のいずれかが成り立つことである.

- (1)  $\varphi_i \in \mathcal{A}$ , または  $\varphi_i \in \Phi$
- (2) ある  $\mathcal{R}$  によって  $\varphi_i$  が有限個の  $\varphi_j (j < i)$  から直接導出可能.

□

$\Phi = \phi$  のとき,  $\vdash \varphi$  と書き,  $\varphi$  は  $S$  の定理 (theorem)  $\mathcal{T}$  という. ある  $\varphi$  に対し  $\varphi \wedge \neg\varphi$  が定理  $\mathcal{T}$  であるとき,  $S$  は矛盾 (inconsistent) するといひ, そうでないとき無矛盾 (consistent) であるといひ.

図 2.1: 形式的体系  $S$ 

## (2) セマンティックス

形式的体系  $S$  において, 解釈  $m$  は論理式集合  $\Phi$  の各論理式の真偽を規定する.

[定義 2.2] (充足可能)  $S$  における論理式  $\varphi$  が真となる  $m$  が存在するとき,  $\varphi$  は充足可能 (satisfiable) という.  $S$  における  $\Phi \vee \varphi$  に対し,  $\Phi$  のすべての論理式が真であるような  $m$  のもとで常に  $\varphi$  が真であるとき,  $\varphi$  は  $\Phi$  の論理的帰結 (logical consequence) であるといひ,  $\Phi \models \varphi$  と書く. □

$S$  における  $\Phi \vee \varphi$  に対し

$\Phi \vdash \varphi$  ならば  $\Phi \models \varphi$  のとき **健全性** (soundness)

$\Phi \models \varphi$  ならば  $\Phi \vdash \varphi$  のとき **完全性** (completeness)

という。  $\varphi$  がすべての解釈に対して真であるとき、  $\models \varphi$  と書き、  $\varphi$  は**恒真式**<sup>1</sup>(valid formula) という。

## 2.2 命題論理

叙述であって、その内容の真偽が一意に定まる文を**命題**という。 **命題論理**とは命題  $p$  の真偽から、それらの関係を論ずる体系である。

[例 2.1] (命題  $p$  の例)

- (1)  $p$ : "人間は死ぬ。"
- (2)  $p$ : "日は東から昇る。"
- (3)  $p$ : "2 は偶数である。"

□

[定義 2.3] 命題論理の形式的体系  $L$  は論理記号、**含意** (implication)  $\supset$ 、**否定** (negation)  $\neg$ 、左かっこ (、右かっこ )、および可算個の**命題変数**  $p_i$  をもち、次の (1),(2) をもつもののみが論理式であるような体系と定義される。

- (1) 命題変数は論理式である。
- (2)  $p, q$  が論理式ならば、  $\neg p, p \supset q$  は論理式である。

□

命題  $p, q$  の真偽の値をそれぞれ真理値 (truth value) T(truth), F(faulte) で表す。ここで、論理記号  $\neg, \supset$  は表 2.2 の真理値表で表される論理式の真偽を意味する。すなわち、この写像が  $L$  の解釈であり論理式の意味である。

表 2.1: 否定  $\neg$ 、含意  $\supset$  の真理値表

$p$	$\neg p$	$p$	$q$	$p \supset q$
T	F	T	T	T
F	T	T	F	F
		F	T	T
		F	F	T

定義で用いられた論理記号より、論理式  $p, q$  に対しさらに次の**論理積** (conjunction)  $\wedge$ 、**論理和** (disjunction)  $\vee$ 、**同値** (equivalence)  $\equiv$  が用いられる。

<sup>1</sup>トートロジー (tautology) ともいう。

ここで、それぞれ

$$\begin{aligned} \neg(p \supset \neg q) \text{ ならば } p \wedge q \\ \neg p \supset q \text{ ならば } p \vee q \\ (p \supset q) \wedge (q \supset p) \text{ ならば } p \equiv q \end{aligned} \quad (2.1)$$

で表す. 表 2.2 に命題論理の形式的体系  $L$  の公理  $\mathcal{A}$ , 推論規則  $\mathcal{R}$  の例を示す.

表 2.2: 公理  $\mathcal{A}$  と推論規則  $\mathcal{R}$

	命題論理 $L$	(第 1 階) 述語論理 $K$
公理 $\mathcal{A}$	(L1) $p \supset (q \supset p)$ (L2) $(p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r))$ (L3) $(\neg p \supset q) \supset ((\neg p \supset \neg q) \supset p)$ (L4) $p \wedge q \supset p$ (L5) $p \wedge q \supset q$ (L6) $p \supset (q \supset p \wedge q)$	(K1) (L1) $\sim$ (L6) (K2) $\forall x P \supset P(t)$ (K3) $P(t) \supset \exists x P$ ( $t$ は項である)
推論規則 $\mathcal{R}$	(RL1) $\vdash p, \vdash p \supset q$ ならば $\vdash q$ (modus ponens)	(RK1) (RL1) (RK2) $P \supset Q(a)$ ならば $P \supset \forall x Q(x)$ (RK3) $P(b) \supset Q$ ならば $\exists x P \supset Q(x)$ (変数 $a, b$ はそれぞれ $P, Q$ には表れないとする)

**[定理 2.1] (推論規則)**  $L$  における論理式  $p, q$  に対し,  $q$  は  $p$  と  $p \supset q$  から直接導出可能である.  $\square$

これは  $p$  と  $p \supset q$  が恒真ならば  $q$  も恒真であるという推論規則で, いわゆる三段論法が真であることを示している. この定理はモーダスポーネンス (modus ponens), または分離規則ともいわれる. これを  $\frac{p, p \supset q}{q}$  とかく.

**[定理 2.2] (演繹定理 (deduction theorem))**  $L$  における  $p, q$ , 論理式集合  $\Phi$  に対して  $\Phi \wedge p \vdash q$  ならば,  $\Phi \vdash p \supset q$  である.  $\square$



[定理 2.3] (標準形定理)  $L$  における命題変数またはその否定  $p_{ij}, q_{ij}$  から生成される論理式  $\varphi$  に対し,  $\varphi$  が恒真でなければ

$$\varphi_1 = \bigvee_{i=1}^n \left( \bigwedge_{j=1}^{m_i} p_{ij} \right) = \bigvee_{i=1}^n (p_{i1} \wedge p_{i2} \wedge \cdots \wedge p_{im_i}) \quad (\text{和標準形}) \quad (2.2)$$

$$\varphi_2 = \bigwedge_{i=1}^n \left( \bigvee_{j=1}^{m_i} q_{ij} \right) = \bigwedge_{i=1}^n (q_{i1} \vee q_{i2} \vee \cdots \vee q_{im_i}) \quad (\text{積標準形}) \quad (2.3)$$

と表現でき,  $\vdash \varphi \equiv \varphi_1, \vdash \varphi \equiv \varphi_2$  となる論理式が存在する.  $\square$

このほか,  $L$  は無矛盾であり, 健全性・完全性が成り立つ. これらは命題論理のメタ定理 (meta-theorem) とよばれる. メタ定理とは議論の対象となっている論理に関する定理である.

### 2.3 述語論理

変数  $x_1, x_2, \dots, x_n$  を含む叙述であって, その変数の値が与えられたとき, 叙述の内容の真偽が定まる文を述語という. 述語論理とは述語  $Q(x_1, x_2, \dots, x_n)$  の真偽を扱う体系である. これを関数論理ともよぶ. 命題は 0 変数の述語である. 変数が述語を表すとき, それを述語変数という. 引数に述語を含まないとき第 1 階述語という.

[例 2.2] (述語  $Q(x_1, x_2, \dots, x_n)$  の例)

- (1)  $Q(x)$ : "太郎は学生である" : 学生 (太郎)
- (2)  $Q(x_1, x_2)$ : "太郎と花子は友達である" : 友達 (太郎, 花子)
- (3)  $Q_1(x_1, Q_2(x_2, x_3))$ : "一郎は太郎と花子が友達であることを知っている" : 知る (一郎, 友達 (太郎, 花子))(第 2 階述語の例)

$\square$

ここでは, 第 1 階述語論理について述べる.

**[定義 2.4]** 述語論理の形式的体系  $K$  は論理記号  $\neg, \supset$ , 全称記号  $\forall$ , 左かっこ  $($ , 右かっこ  $)$ , カンマ  $(,)$ , および可算個の**個体変数** (individual variable)  $x_i$ , 可算個の**述語** (predicate symbol)  $Q_j(\cdot)$  と可算個の**関数**  $f_k(\cdot)$  をもち, 次の (1) ~ (4) のみが論理式であるような体系と定義される.

- (1) 個体変数  $x_i$  は**項** (term) である.
- (2)  $f_k$  が  $n$  変数関数,  $t_1, t_2, \dots, t_n$  が項ならば,  $f_k(t_1, t_2, \dots, t_n)$  は項である.
- (3)  $Q_j$  が  $n$  変数述語,  $t_1, t_2, \dots, t_n$  が項ならば,  $Q_j(t_1, t_2, \dots, t_n)$  は論理式である.
- (4)  $x$  が変数,  $P, Q$  が論理式ならば  $\neg P, P \supset Q, \forall x P$  はいずれも論理式である.

ここで, 0 変数関数は**個体定数** (individual constant) といわれる. また, 0 変数述語は**命題**である. □

個体変数は“もとの世界”のもの (対象) に対応する.  $n$  個の個体変数をもつ述語  $Q(t_1, t_2, \dots, t_n)$  を**基本論理式** (atomic formula) とよぶ.

個体変数を  $x$ , 論理式を  $Q$  とし,  $\forall x Q(x)$  あるいは  $\exists x Q(x)$  において, 変数  $x$  を  $\forall x, \exists x$  による**束縛** (bound) **変数**という. 束縛されていない変数を**自由** (free) **変数**という. ここで,  $\exists$  は**存在記号**,  $\forall, \exists$  両者を**限定記号**とよぶ. 限定記号  $\forall, \exists$  は,  $\neg(\forall x(\neg Q(x)))$  ならば  $\exists x Q(x)$  を表す. 表 2.2 に述語論理の形式的体系  $K$  の公理  $\mathcal{A}$ , 推論規則  $\mathcal{R}$  の例を示す.

**[定理 2.4]** (**一般化規則** (generalization rule))  $K$  における論理式  $P, Q$ , 変数  $x$  に対して  $P \supset (\forall x Q)$  は,  $P \supset Q$  から直接導出可能である. □

述語論理においては, 論理式の解釈を対象定義領域  $\mathcal{D}$  と付値で定義する.  $\mathcal{D}$  は任意の空でない一つの集合である.

付値は,  $L$  における  $\{0, 1\}$  の対応と同様, 自由変数  $x_i$  に対し  $\mathcal{D}$  の要素を対応させる. すなわち,  $x_1, x_2, \dots, x_n$  に対し  $\mathcal{D}^n$  から  $\{0, 1\}$  へ写像する.

[定義 2.5] (解釈) 空でない定義域  $\mathcal{D}$  の元により,  $n$  変数関数  $f_k(\cdot)$  に対する  $n$  変数述語  $Q_j(\cdot)$  に  $\mathcal{D}^n$  から  $\{0,1\}$  への写像  $M$  を解釈  $m$  という.

$$m : Q_j \in \mathcal{D}^n \rightarrow M(Q_j) \in \{0,1\}. \quad (2.4)$$

ただし, 個体定数  $c$  と  $n$  変数関数  $f_k(\cdot)$  に対し

$$M(c) \in \mathcal{D} \quad (2.5)$$

$$f_k \in \mathcal{D}^n \rightarrow M(f_k) \in \mathcal{D} \quad (2.6)$$

である. □

一般に, 解釈  $m$  の与え方は無限に存在する. どのような  $m$  でも  $M(\cdot) = 1$  となる論理式は恒真式である. 論理式集合  $\Phi$  のすべての論理式が解釈  $m$  のもとで真のとき,  $m$  を  $\Phi$  の**モデル** (model) という.

[定理 2.5]  $K$  の論理式集合  $\Phi$  が完全であるというのは, 任意の論理式  $\varphi$  に対して  $\Phi \vdash \varphi$  であるか否かが決定可能であるとき, かつそのときに限る. □

$\Phi$  が自然数論の公理を含むならば  $\Phi$  は完全ではないことが知られている (Gödel の不完全定理).

$K$  における論理式  $\varphi$  に対し  $\varphi$  が恒真でなければ, 次の形の論理式  $\vdash \varphi \equiv \varphi_1, \vdash \varphi \equiv \varphi_2$  となる  $\varphi_1, \varphi_2$  が存在する.

$$\varphi_1 = \Gamma x_1, \Gamma x_2, \dots, \Gamma x_k \bigvee_{i=1}^n (p_{i1} \wedge p_{i2} \wedge \dots \wedge p_{im_i}) \quad (2.7)$$

$$\varphi_2 = \Gamma x_1, \Gamma x_2, \dots, \Gamma x_k \bigwedge_{i=1}^n (q_{i1} \vee q_{i2} \vee \dots \vee q_{im_i}) \quad (2.8)$$

ここで,  $\Gamma$  は限定記号  $\forall$  か  $\exists$  であり,  $p_{ij}, q_{ij}$  は基本論理式, またはその否定である. これを**標準形定理**という. 命題論理におけると同様に, **メタ定理**が  $K$  に対しても成り立つ.

### 演習問題

[2.1]  $\{\exists x \forall y F(x, y)\} \supset \{\forall y \exists x F(x, y)\}$  を証明し, また, 例をあげて説明せよ.

[2.2] (1)  $A \supset B, A \supset (B \supset C)$  より,  $A \supset C$  を導く証明図を示せ.

(2)  $(A \supset (B \supset C)) \supset ((A \wedge B) \supset C)$  を証明せよ.

[2.3] “ $x$  は素数である” :  $\text{Prime}(x)$  を論理記号を用いて表せ.

[2.4] 命題論理において, 前提を  $p$ , 結論を  $q$  として,  $p \models q$  (論理的帰結) を背理法を用いて証明する. これを論理式を用いて示せ.

### 参考文献

1. 前原昭二, 記号論理入門, 日評数学選書, 日本評論社
2. 入江盛一, 数理論理学入門, 培風館, 1973 年.
3. 赤堀也, 伊藤正夫, 後藤英一, 広瀬健編, 情報処理のための数学, 共立出版, 昭和 50 年.
4. 宮川洋, 情報論 —情報処理の理論—, 岩波全書, 1979 年.
5. 林晋, 数理論理学, コロナ社, 1989 年.
6. 小倉久和, 高濱徹行, 情報の論理数学入門, 近代科学社, 1991 年.
7. 道脇義正, 情報科学のための数学入門, 東京図書, 1991 年.
8. 赤間世紀, 計算論理学入門 AI とコンピュータサイエンスへの論理的アプローチ, 哲学出版株式会社, 1992 年.
9. 小野寛晰, 情報科学における論理, 日本評論社, 1994 年.
10. 田中尚夫, 計算論理学入門, 裳華房, 1997 年.

# 3

## オートマトンと言語

オートマトン (automaton) はデカルト (R.Descartes) が初めて用いた言葉といわれ、ギリシャ語の auto(自動) と matos(動くもの) を合成した自動機械を意味する。1943年, W.S.McCulloch と W.Pitts は生体の神経網をモデル化した。神経系はニューロンの集合でモデル化され、それらの接続により複雑な論理活動を行う。1956年, S.C.Kleene はこれと同等な抽象的数学モデルを用いてその能力を解明している。その結果、最も簡単な有限の記憶量をもつ情報処理機械として有限状態オートマトンの正則性の概念を明らかにしている。

一方、1959年 N.Chomsky は数学的形式文法である句構造文法 (phrase structure grammar) を導入した。形式言語 (formal language) はオートマトンを用いて言語を生成する形式文法により規定される。オートマトンはいわばコンピュータの単純なモデルであり、形式言語はこれで受理 (認識) される入力系列である。

### 3.1 有限状態オートマトン

離散的な時刻  $t$  において、記号  $a \in \Sigma$  を入力とし、過去の入力の履歴を有限個の内部状態  $q \in Q$  で記憶し、これに依存した記号  $b \in P$  を出力する順序機械を有限状態オートマトン (finite state automaton : FA) という。集合  $\Sigma, Q, P$  を2元記号で表すとき順序論理回路で表現される。

各時刻において、入力記号列に対して状態遷移が一意に定まるとき決定性有限状態オートマトン (deterministic FA : DFA)、そうでないとき非決定性有限状態オートマトン (nondeterministic FA : NFA) という。

[定義 3.1] (有限状態オートマトン) 状態の有限集合を  $Q$ 、入力記号の有限集合を  $\Sigma$ 、状態遷移関数を  $\delta(\cdot, \cdot)$ 、初期状態を  $q_0 \in Q$ 、最終状態集合を  $F \subseteq Q$ 、とするとき、有限状態オートマトン  $A$  を

$$A = \langle Q, \Sigma, \delta, q_0, F \rangle \tag{3.1}$$

で定義する。ここで、ある時刻  $t$  で状態が  $q \in Q$ 、入力記号が  $a \in \Sigma$  のとき、時刻  $t + 1$  において  $\delta(q, a)$  に遷移する。□

出力記号の有限集合を  $P$ , 出力関数を  $r : Q \rightarrow P$  とするとき, 出力記号  $b \in P$  は

$$b = r(q) \quad (3.2)$$

で与えられる. このとき, 有限状態オートマトン  $A'$  を  $A' = \langle Q, \Sigma, P, \delta, r, q_0, F \rangle$  と定義するが,  $b$  は  $q$  により一意に決定されるから  $A$  は  $A'$  と等価である.

$\Sigma$  よりとり出した有限長の記号列  $x$  を  $\Sigma$  上の**語** (word), または**文** (sentence) という.  $\Sigma$  上のすべての語の集合を  $\Sigma^*$  で表す. 長さが 0 の記号列<sup>1</sup>  $\lambda \in \Sigma^*$  を**空系列** (empty sequence) または**空文** という. ここで,  $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$  とおく.  $\Sigma$  上の語の任意の無限集合  $L \subset \Sigma^*$  が**形式言語** または単に**言語** (language) である.

語  $x$  の次に語  $y$  を並べた文字列  $xy$  は  $x, y$  の**接続** (concatenation) という.

### (1) 受理集合

定義 3.1 における写像  $\delta : Q \times \Sigma^* \rightarrow Q$  は  $x \in \Sigma^*, a \in \Sigma$  に対し

$$\delta(q, xa) = \delta(\delta(q, x), a) \quad (3.3)$$

である. 有限状態オートマトン  $A$  が記号列  $x$  を**受理** (accept) するとは,  $x \in \Sigma^*$  に対し

$$\delta(q_0, x) \in F \quad (3.4)$$

が成立することである.  $A$  の受理集合, すなわち  $A$  が受理する記号列の集合を  $L(A)$  と書く. なお, 受理することを**認識** (recognize) するともいう.

### (2) 状態遷移

有限状態オートマトンは**状態遷移図** (state graph) で記述される. すなわち時刻  $t$  において, 入力  $a^t \in \Sigma$ , 状態  $q^t \in Q$ , 出力  $b^t \in P$  とするとき, 図 3.1 のようなラベルつき有向グラフで示す. 図 3.1 で  $q^{t+1} \in F$  のとき, これを二重丸で示し, 出力  $b^t$  を規定しないとき, これを省略する.

受理する入力記号列と状態遷移図の関係を図 3.2 に示す. すべての有限状態オートマトンは, 図 3.2 の 3 つのいずれかに分解できることが知られている. 逆に, これらを組み合わせて受理する記号列の集合がわかり, 任意の二つの有限状態オートマトンの等価性を明らかにすることができる.

<sup>1</sup>記号を 1 つも含まない系列.

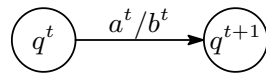


図 3.1: 有限オートマトンの状態遷移図

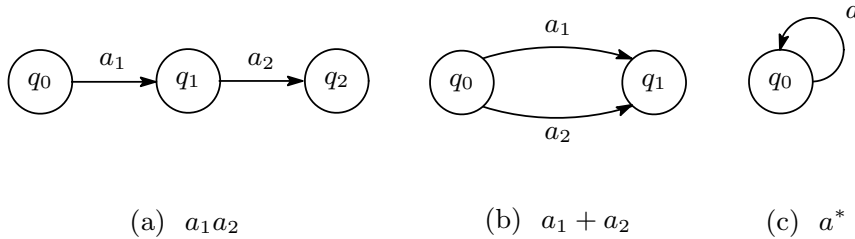
(a)  $a_1a_2$ (b)  $a_1 + a_2$ (c)  $a^*$ 

図 3.2: 記号列と状態遷移図

なお、任意の非決定性有限状態オートマトンに対して、それと等価な決定性有限状態オートマトンが存在すること、任意の有限状態オートマトンに対して最簡型とよばれる等価で状態数が最小な決定性有限状態オートマトンが存在することが知られている。

### (3) 正則性

Kleene は有限状態オートマトンによって受理される集合は**正則集合** (regular set) で、逆も成立することを示した。これを **Kleene の定理** という。正則集合は、次の**正則表現** (regular expression) の表す記号列の集合で定義される。

[定義 3.2] (正則表現) 次に規定されるもののみが**正則表現**である。

- (1)  $a \in \Sigma$ ,  $\lambda$ , 空集合  $\phi$  はそれだけで正則表現である。
- (2)  $\alpha, \beta$  が正則表現のとき,  $(\alpha\beta), (\alpha + \beta), (\alpha^*)$  も正則表現である。ここで,  $\alpha\beta$  は接続を,  $\alpha + \beta$  は  $\{\alpha, \beta\}$  を,  $\alpha^*$  は**クリーネ閉包** (closure) を示す。□

正則表現  $\alpha$  の正則集合を  $[\alpha]$  とすると  $[\alpha] = \{\alpha\}$ , 長さ 0 の記号  $\lambda$  よりなる系列をただ一つ含む集合  $[\lambda] = \{\lambda\}$ , なんらの系列を含まない集合  $[\phi] = \phi$ , また  $[\alpha\beta] = [\alpha] \cdot [\beta]$ ,  $[\alpha + \beta] = [\alpha] \cup [\beta]$ ,  $[\alpha]^0 = [\lambda]$ ,  $[\alpha]^1 = [\alpha]$ ,  $[\alpha]^{n+1} = [\alpha]^n \cdot [\alpha]$ ,  $[\alpha^*] = [\alpha]^*$  である。

ここで、任意の正則表現  $\alpha, \beta, \gamma$  に対し次式が成り立つ.

$$\begin{aligned}
 \lambda\alpha &= \alpha\lambda = \alpha \\
 \phi\alpha &= \alpha\phi = \phi \\
 \lambda^* &= \phi^* = \lambda \\
 \alpha(\beta\alpha)^* &= (\alpha\beta)^*\alpha \\
 (\alpha + \beta)^* &= (\alpha^*\beta^*)^* \\
 (\alpha^*\beta)^* &= \lambda + (\alpha + \beta)^*\beta \\
 \alpha(\beta + \gamma) &= \alpha\beta + \alpha\gamma \\
 \alpha^* &= \lambda + \alpha\alpha^* = \lambda + \alpha + \alpha^2 + \cdots + \alpha^n + \cdots
 \end{aligned} \tag{3.5}$$

定義 3.2 (2) は、 $\alpha, \beta$  を受理する有限状態オートマトンに対し、 $\alpha\beta$  は**直列結合**、 $\alpha + \beta$  は**並列結合**、 $\alpha^*$  は**フィードバックループ**をもつ有限状態オートマトンの存在を示し、受理できる系列の拡張を与えている (図 3.2 参照).

### 3.2 形式言語

言語とは単語の集合と、それらの単語からなるあらゆる正しい文章 (文法にしたがった文章) の集合と考える. 文章の意味を考えないで単語と単語の結合の仕方だけを考える. このように構文だけに注目した言語が形式言語である. すなわち、明確に言語の数学的モデルを定め、形式的に規定された言語を形式言語とよぶ. 言語を受理するオートマトンを示すことにより、その言語を生成する形式文法を規定する. 言語とオートマトン、チューリングマシンの関連は、1941 年の A.Thue による準チューシステムにさかのぼる.

#### (1) 形式文法

有限状態オートマトン  $A$  による  $\Sigma$  上の形式言語  $L(A)$  は、 $A$  が受理、あるいは認識する  $\Sigma^*$  の中にある特定の条件を満足する記号列、すなわち文  $x$  だけからなる部分集合  $L(A)$  として次のように定められる.

$$L(A) = \{x \in \Sigma^* \mid \delta(q_0, x) \in F\} \tag{3.6}$$

ここで、 $A$  を**正則オートマトン**という.

#### (2) 生成規則

文章の構文を構文木で表現すると、語は木の終端の葉になる. したがって、言語では語の集合  $\Sigma$  は**語彙** (vocabulary) であり、これを**終端記号** (terminal



symbol), または定数とよぶ. 一方, 構文木の枝は構文カテゴリ名を示しこれを**非終端記号**<sup>2</sup> (nonterminal symbol), または構文変数 (syntactic variable) とよび, その集合を  $N$  で表す. 構文木の根は文生成の出発点であり, これを**開始記号** (start symbol) とよび, その集合を  $S$  で表す.  $N$  は自然言語文法における句, 節, 文のような高位の抽象概念であり, 文の生成方法を示す. したがって,  $N$  は文構造の中間段階を示し外部構造には表れない.  $N$  を一つも含まない文形式を**文**という.

さらに, 言語の文法規則を**生成規則** (production rule) あるいは**書換え規則** (rewriting rule) とよび, これを  $R$  で表す. その結果,  $S \subseteq N$  を  $R$  の規則で書換える生成文法, あるいは形式文法  $G$  は

$$G = \langle N, \Sigma, R, S \rangle \quad (3.7)$$

で定義される.  $R$  による記号列の書換え  $\varphi \rightarrow \theta$  を次のように示す. 生成文法  $G$  において, 記号列  $\alpha, \beta, \varphi, \theta \in V^*$  に対し,  $(\varphi \rightarrow \theta) \in R$  ならば  $\alpha\varphi\beta \xrightarrow{G} \alpha\theta\beta$  と書き,  $\alpha\varphi\beta$  は  $\alpha\theta\beta$  を**直接導出**する, または**書換え**られるという. ここで,  $\varphi \neq \lambda, V = N \cup \Sigma$  ( $N \cap \Sigma = \emptyset$ ),  $V^+ = V^* \setminus \{\lambda\}$  である.  $n = 1, 2, \dots$ , に対し,  $\gamma_1 \xrightarrow{G} \gamma_2 \xrightarrow{G} \gamma_3 \cdots \xrightarrow{G} \gamma_{n+1}$  ならば,  $\gamma_1 \xrightarrow{*G} \gamma_{n+1}$ , または  $\gamma_1, \gamma_{n+1} \in V^*$  に対し,  $\gamma_1 \xrightarrow{G} \gamma_{n+1}$  と書き,  $\gamma_1$  は  $\gamma_{n+1}$  を**導出**するという.  $S \xrightarrow{*G} x$  となる  $x$  は文形式である. すなわち文法  $G$  によって生成される言語  $L(G)$  は

$$L(G) = \{x \mid x \in \Sigma^*, S \xrightarrow{*G} x\} \quad (3.8)$$

と書き表される. なお, 語の列  $(\varphi, \theta)$  の集合を辞典  $D$  ということがある.  $\theta$  を  $D$  による  $\varphi$  の解釈という. Thue の語の問題とは,  $\gamma_1, \gamma_{n+1} \in \Sigma^+$  に対し  $\gamma_{n+1}$  が  $\gamma_1$  から導出されるか否かを決定する有限的方法が存在しているかという問題で, 準チューシシステムとして E.L.Post, A.A.Markov によって考察されている.

### (3) 形式言語の文法

Chomsky による言語階層は, 次に示す生成規則としての文法の階層概念 (Chomsky hierarchy) により定義される.

---

<sup>2</sup>中間記号ともいう.

[定義 3.3] (Chomsky の言語階層) 生成規則  $R$  が次の条件  $0, 1, \dots, i$ , を満たすとき, type  $i$  文法 ( $i = 0, 1, 2, 3$ ) という.

(条件 0)  $\alpha\varphi\beta \rightarrow \alpha\theta\beta$ ,  $\varphi \in V^*NV^*$ ,  $\alpha, \beta, \theta \in V^*$

(条件 1)  $\alpha\varphi\beta \rightarrow \alpha\theta\beta$ ,  $\alpha, \beta \in V^*$ ,  $\varphi \in N, \theta \in V^+$

(条件 2)  $\varphi \rightarrow \theta$ ,  $\varphi \in N$ ,  $\theta \in V^*$

(条件 3)  $\varphi \rightarrow \theta$ , または  $\varphi \rightarrow \theta\theta'$ ,  $\varphi, \theta' \in N$ ,  $\theta \in \Sigma$  □

これらは文法の複雑さを表す一つ概念であり,  $i$  が大きいほど単純であり水準が低い. また言語  $L$  が type  $i$  文法で生成されるとき,  $L$  を type  $i$  言語という.

#### (4) 言語階層

Kleene の定理により正則集合は有限状態オートマトンにより受理, すなわち認識される. これは type 3 文法, または正則文法で生成できる集合と一致する. 有限状態オートマトンの能力は, 第 4 章で述べるチューリングマシンの能力に比べかなり劣る. すなわち, 認識機械としてみると前者は正則集合のクラスしか認識できないが, 後者は type 0 文法で生成できる集合, すなわち帰納的可算集合のクラスを認識する. 両者の間には言語階層によるタイプにより, 次の階層的部分クラスが存在する.

- ① **句構造文法** (phrase structure grammar) どんな制約もおかれない最も一般的な文法で無制限文法ともいう. 少なくとも一つの非終端記号を含む語を書換える.
- ② **文脈依存文法** (context-sensitive grammar) 条件 1 は,  $\varphi \rightarrow \theta$  の書換えが文脈  $\alpha, \beta$  においてのみ可能であることを意味する.  $\theta \neq \lambda$  である.  $|\varphi| \leq |\theta|$  で定義する<sup>3</sup>ことがあるが, 本質的に同一である. 文脈規定文法ともいう.
- ③ **文脈自由文法** (context-free grammar) 条件 2 は文脈に依存しないことを示す. すなわち, 単一の非終端記号を書換える.
- ④ **正則文法** (regular grammar) 条件 3 は線形であることを示す. これを左線形文法という.  $\varphi \rightarrow \theta$ , または  $\varphi \rightarrow \theta'\theta$  のとき右線形文法という. 制限が最も強い文法である. 正則言語は他の文法でも生成される. しか

<sup>3</sup> $|x|$  は記号列  $x$  の長さを示す.

し、他の言語の中には正則文法で生成できないものがある<sup>4</sup>。

言語を分類すると、表 3.1 のように示すことができる。ここで、包含関係は  $L_i \supseteq L_{i+1}$  ( $i = 0, 1, 2$ ) である。

認識機械との対応は、コンパイラなど言語処理プログラム、特に構文解析などの設計には重要である。ALGOL などのコンパイラはおおよそ type 2 の文法に従っていると考えられる。

表 3.1: Chomsky による言語階層

生成文法	言語型	言語名	認識機械	備考
type 0	$L_0$	句構造言語	チューリングマシン	帰納的可算集合
type 1	$L_1$	文脈依存言語	線形有界オートマトン	
type 2	$L_2$	文脈自由言語	プッシュダウンオートマトン	
type 3	$L_3$	正則言語	有限状態オートマトン	正則集合

## 演習問題

[3.1] 入力系列 (1)  $(1 + 000 + 001)^*1$ , (2)  $(0 + 10^*10^*1)^*10^*10^*1$  を受理するオートマトンの遷移図を示せ。ただし、(1),(2) の正則表現において  $x^*$  は、 $x^0 = \lambda$ ,  $x^n (n \geq 1)$  のとき  $x$  の  $n$  個の連を示す。また、ここでオートマトンが受理するとは、(1),(2) の入力系列に対し出力に 1 が生じることを意味する。

[3.2]  $a, b, c$  を変数,  $+, \times, (, )$  を演算子とする数式を生成する文法  $G$  を示せ。

## 参考文献

1. 入江盛一, 数理論理学入門, 培風館, 1973 年.
2. 赤堀也, 伊藤正夫, 後藤英一, 広瀬健編, 情報処理のための数学, 共立出版, 昭 50 年.
3. 宮川洋, 情報論 —情報処理の理論—, 岩波全書, 1979 年.
4. 米田政明, 計算機科学の基礎, 森北出版株式会社, 1991 年.
5. 長尾真, 言語工学, 昭晃堂, 昭 58 年.
6. 田中尚夫, 計算論理学入門, 裳華房, 1997 年.
7. 北研二, 確率的言語モデル, 東京大学出版会, 1999 年.

<sup>4</sup>例えば、 $\{a^n b^n (n = 1, 2, \dots)\}$  は正則文法では生成できず、文脈自由文法が必要であること、 $\{a^n b^n c^n (n = 1, 2, \dots)\}$  は文脈自由文法では生成できず、文脈依存文法が必要であることが知られている。



## 4 チューリングマシンと計算論

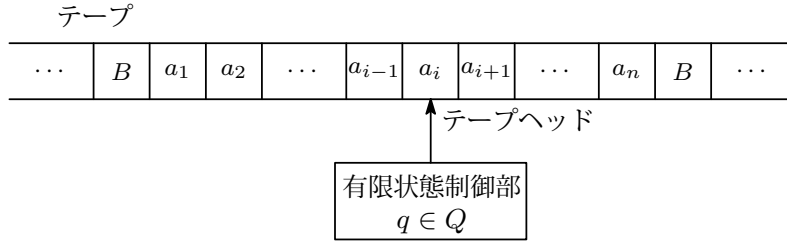
1936年、A.M.Turingは計算の理論を構築するために抽象的な計算機械であるチューリングマシン  $T_m$  を考案した。 $T_m$  を用いて複雑な計算も単純な操作の組み合わせで構成できることを示し、与えられた問題を解くアルゴリズムが存在するか否かを明快に説明しようとするものである。チューリングマシンはいくつかの変形がある。実際、任意の計算を実行する万能チューリングマシンの構成と動作原理はノイマン型コンピュータの動作サイクルと対応する。なお、チューリングマシン  $T_m$  は3.2節に述べたとおり、Chomskyによる言語階層で無制限言語  $L_0$  の受理系である。

また、有限状態オートマトンは正則性の概念より明らかな通り、最も簡単な情報処理機械であった。その意味でチューリングマシンは強力で、有限状態オートマトンはこれよりかなり能力が劣る。

一方、チューリングマシン  $T_m$  が考案される以前に、1934年 J.Herbrand, K.Gödel, S.C.Kleeneらは帰納的関数 (recursive function) を導入している。また、同時期に A.Churchによるラムダ計算もある。これらの概念はいずれも独立に考えられ形式も異なっているが、本質的には同値な計算機械モデルであり、いずれも計算可能関数 (computable function) を定義するために用いられる。実際に計算可能な関数は帰納的関数であるという主張を Churchの提唱という。現在までこの提唱は妥当であるという確信をくつがえすような証拠はなく、広く認められている。また、チューリングマシンで計算できる関数の集合は帰納的関数の集合と一致することを示すことができる。なお、上記の互いに同値な3つの計算機械モデルを超える計算あるいは計算機械モデルは考え出されていない。

### 4.1 チューリングマシン

チューリングマシン  $T_m$  は図4.1に示すように、テープ・テープヘッド・有限状態制御部、よりなる。テープは一定区画のコマに分割されており、両方向に無限長である。ただし、有限個のコマ以外は空白記号  $B$  で埋まっているものとする。テープヘッドはコマの中の記号の読出し・書込みを行う。有限状態制御部はその内部状態とヘッドの読出し記号により定まる次の動作を制御する。その結果、次の状態への遷移・テープへの記号書込み・テープヘッドの移動、が行われる。

図 4.1: チューリングマシン  $T_m$ 

〔定義 4.1〕 (チューリングマシン) 状態の有限集合を  $Q$ , テープ記号の有限集合を  $T$ , 次動作関数を  $\delta(\cdot, \cdot)$ , 空白記号を  $B \in T$ , 初期状態を  $q_0 \in Q$ , 最終状態集合を  $F \subseteq Q$  とするとき, チューリングマシン  $T_m$  を

$$T_m = \langle Q, T, \delta, B, q_0, F \rangle \quad (4.1)$$

で定義する. ここで,  $\delta(\cdot, \cdot)$  は現在の状態を  $p \in Q$ , テープヘッドの読出し記号を  $a \in T$  とし, 次の状態を  $q \in Q$ , テープヘッドの書込み記号を  $b \in T$ , テープヘッドの移動を  $d \in \{L, H, R\}$  とするとき

$$\delta(p, a) = (q, b, d) \quad (4.2)$$

で表される.  $L \cdot H \cdot R$  はそれぞれヘッド移動を, 左へ1コマ・保持・右へ1コマ, を示す.  $\delta(\cdot, \cdot)$  は命令ともよばれる.  $\square$

なお,  $p_f \in F$  とするとき, 任意の  $a \in T$  に対し,

$$\delta(p_f, a) = (p_f, a, H) \quad (4.3)$$

とし, チューリングマシンは停止する.

また, チューリングマシンにおいて  $\delta(p, a)$  が2つ以上あるとき, **非決定性チューリングマシン**といい非決定的にいずれかを選ぶものとする. そうでなくただか一つの  $\delta(\cdot, \cdot)$  をもつとき, **決定性チューリングマシン**という. 任意の非決定性  $T_m$  に対し同一の受理集合をもつ決定性  $T_m$  の存在することが知られている.

#### (1) 受理機械と計算機械

$\Sigma \subseteq T \setminus \{B\}$  とし  $a = (a_1, a_2, \dots, a_n) \in \Sigma^*$  とする. ただし,  $\Sigma^*$  は  $\Sigma$  上のすべての語の集合である.  $a$  を  $T_m$  のテープ上に書き込み, 初期状態を  $q_0$  とし動作を開始する. 最後に状態  $p_f \in F$  で停止するとき,  $a$  は  $T_m$  によって受

理されるという。すなわち、 $T_m$  は  $a \in \Sigma^*$  の受理機械 (認識機械ともいう) である。

一方、 $T_m$  は開始時点のテープの内容を入力とし、停止したときのテープの内容を出力とする**計算機械**でもある。

## (2) 万能チューリングマシン

チューリングマシン  $T_m$  の変形として次のようなものが考えられる。

- (1) 片側無限テープのもの
- (2) 多テープのもの
- (3) 多ヘッドのもの

いずれも計算可能性からみると互いに等価なことが示され、したがって、 $T_m$  は計算機械として十分に一般的であることが示されている。

さて、万能チューリングマシンを  $U$  としよう。  $U$  は任意のチューリングマシン  $T_m$  を実現する。すなわち、 $U$  は任意の  $T_m$  の命令集合の記述とその入力テープを与えたとき、 $T_m$  が出力すべきテープを出力するような動作をするチューリングマシンである。ここで、 $U$  の動作は、状態・読出し記号、に対する命令をみつけ解釈するインタプリタと考えることができる。通常、処理が複雑になれば、 $T_m$  の状態数は増大する。しかし、有限の状態数の  $U$  は他の任意の状態数の  $T_m$  で行える動作をすべて実行することができるのである。ただし、 $U$  はほとんどあらゆる  $T_m$  より能力は劣る<sup>1</sup>。これは専用の  $T_m$  と汎用の  $U$  とのちがいであり、ハードウェアとソフトウェアの関係を示している。

## 4.2 計算論

### (1) 計算可能性

原始的で単純な機能しかもたない、しかしチューリングマシンが計算できる自然数上の関数を定義する。いま、 $1 \in T$  とし、非負整数  $k$  を  $\bar{k} = \underbrace{11 \cdots 1}_{k+1}$  で表す。テープ上の  $n$  個の非負整数  $(k_1, k_2, \dots, k_n)$  を  $(k_1, k_2, \dots, k_n) = \bar{k}_1 B \bar{k}_2 B \cdots B \bar{k}_n$  で表す。

<sup>1</sup>万能という言葉はすべてにおいて優るという意味ではない。

**[定義 4.2] (計算可能関数)** 初期状態  $q_0 \in Q$ , テープの内容を  $(x_1, x_2, \dots, x_n)$ , 有限回の動作後最終状態のテープの内容が  $\bar{k}$  であるとき, 非負整数上の  $n$  変数**計算可能部分関数** (partical function)  $\varphi(\cdot)$  は

$$\varphi(x_1, x_2, \dots, x_n) = k \quad (4.4)$$

で定義される. ただし,  $(x_1, x_2, \dots, x_n)$  が  $\varphi$  の定義域に入らなければ  $T_m$  は停止せず,  $\varphi(\cdot)$  は未定義とする. また,  $T_m$  が  $(x_1, x_2, \dots, x_n)$  の全域に対し停止するとき,  $\varphi$  を**全域関数**といい, **計算可能関数**ともいう.  $\square$

計算可能 (部分) 関数はたかだか可算無限個しか存在しない. 数論的関数  $f(\cdot) = \varphi(\cdot)$  のとき, すなわち両者の値が定義され等しく, かつ定義域が一致し  $T_m$  が存在するとき,  $f(\cdot)$  は計算可能であるという.



[定義 4.3] (原始帰納関数) 非負整数上の基本関数

$$(1) \text{ 後者関数 (successor function) : } S(x) = x + 1 \quad (4.5)$$

$$(2) \text{ 射影関数 (projection function) : } U_i^n(x_1, x_2, \dots, x_i, \dots, x_n) = x_i (1 \leq i \leq n) \quad (4.6)$$

$$(3) \text{ 零関数 (zero function) : } Z(x) = 0 \quad (4.7)$$

に次の操作 (4), (5) を有限回繰り返して得られる関数を**原始帰納的関数** (primitive recursive function) という。

$$(4) \text{ 合成 (composition) : } m \text{ 変数関数 } g \text{ と } m \text{ 個の } n \text{ 変数関数 } h_1, h_2, \dots, h_m \text{ から新しい } n \text{ 変数関数 } f(x_1, x_2, \dots, x_n) = g(h_1(x_1, x_2, \dots, x_n), \dots, h_m(x_1, x_2, \dots, x_n)) \text{ をつくる.}$$

$$(5) \text{ 原始帰納法 (primitive recursion) : } n \text{ 変数関数 } g \text{ と } (n + 2) \text{ 変数関数 } h \text{ から新しい } (n + 1) \text{ 変数関数 } f \text{ を次式で定義する.}$$

$$f(x_1, x_2, \dots, x_n, 0) = g(x_1, x_2, \dots, x_n) \quad (4.8)$$

$$f(x_1, x_2, \dots, x_n, y + 1) = h(x_1, x_2, \dots, x_n, y, f(x_1, x_2, \dots, x_n, y)) \quad (4.9)$$

さらに、次の (6) 最小化の操作も許し、それらの有限回の適用で得られる関数を**部分帰納的関数** (partial recursive function) という。

$$(6) \text{ 最小化 (minimization) : } (n + 1) \text{ 変数関数 } g \text{ に対して、新しく } n \text{ 変数関数}$$

$$\exists z, f(x_1, x_2, \dots, x_n) = \min\{z : g(x_1, x_2, \dots, x_n, z) = 0\} \quad (4.10)$$

を定義する。ここで、式 (4.10) の  $z$  が存在しないとき、 $f(\cdot)$  は未定義とする。特に、 $f(x_1, x_2, \dots, x_n)$  が全域的であるとき、(一般) **帰納的関数** (general recursive function) という。□

帰納的関数は自然数から自然数への計算可能な関数である。(1) ~ (3) のような基本計算関数に関数の合成や最小化の操作を繰り返し適用して得られる関数が部分帰納的関数である<sup>2</sup>。チューリングマシンによって計算可能であることと自然数から自然数への部分関数が帰納的であることは同値である。すなわち、計算可能性と帰納性に関して、計算可能 (部分) 関数は (部分) 帰納的関数であ

<sup>2</sup>たとえば、 $f(0) = 1, f(1) = 1, f(n + 2) = f(n + 1) + f(n)$  で表される関数  $f(\cdot)$  は原始帰納的関数で Fibonacci 系列を生成する。

り、またその逆も成立することが知られている。

## (2) 決定問題と停止問題

(a) **決定問題** 集合  $S$  上の決定問題 (decision problem)  $P$  とは  $P : S \rightarrow \{\text{Yes}, \text{No}\}$  で表される。ここで

$$P = \begin{cases} \text{可解 (solvable),} & P \text{ を解くアルゴリズム } A \text{ が存在する;} \\ \text{非可解 (unsolvable),} & P \text{ を解くアルゴリズム } A \text{ が存在しない} \end{cases}$$

という。例えば、“与えられた数が素数か否か (素数判定問題)”は可解である。しかし、非可解な決定問題もある。代表的なものがチューリングマシンの停止問題である<sup>3</sup>。

(b) **停止問題**  $x \in T^*$  を入力とし、 $y \in T^*$  を出力するチューリングマシン  $T_M$  は  $M : x \in T^* \rightarrow y \in T^*$  で一般に部分写像である。このとき、 $x \in T^*$  を入力し

- (i)  $y = M(x)$  となる出力を出して停止する : Yes
  - (ii) 出力を出さないで停止、または停止しない : No
- とするとき

$$P : S \rightarrow \{\text{Yes}, \text{No}\} \quad (4.11)$$

すなわち、 $T_M$  が入力  $x$  に対し“最終状態に遷移し停止するか否か”という決定問題を**チューリングの停止問題** (halting problem) といい、この決定問題  $P$  は非可解であることが知られている<sup>4</sup>。したがって、非可解な決定問題とはアルゴリズムが存在しない問題であり、いいかえれば全域的チューリングマシンが存在しない問題である。もっと基本的には、必ず停止するプログラムが存在しない問題である。

## (3) 抽象的計算機械モデルと計算量

計算量の下界を求めるためにはアルゴリズムを実行する抽象的計算機械モデルが必要である。チューリングマシンのほかにいくつかのモデルが考えられている。たとえば、1本の入力テープと  $(k-1)$ 本の作業用テープをもつ  $k$ テープチューリングマシンが使われる。その中で  $k$ テープ交互チューリングマシン、 $k$ テープ非決定性チューリングマシンは並列コンピュータのモデル、 $k$ テ

<sup>3</sup>ある決定問題が非可解であることは、その問題がチューリングマシンの停止問題に帰着させることにより証明される。

<sup>4</sup>どのようなアルゴリズムでも計算できないような数論的関数が存在することにより証明される。

決定性チューリングマシンは逐次コンピュータのモデルである。また、**ランダムアクセスマシン** (Random Access Machine : RAM) は、読取り専用入力テープ・書込み専用出力テープ・記憶装置・アキュムレータ・プログラム、からなる。その命令は現実のコンピュータのように、算術命令・間接番地・分岐命令などがある。RAMのプログラムはこれらの命令の列で、プログラム自体を記憶装置に格納することはできない<sup>5</sup>。プログラムを記憶装置(レジスタ)に格納可能としたモデルに、**ランダムアクセスプログラム内臓マシン** (Random Access Stored Program machine : RASP) がある。プログラムは実行中に自分自身に含まれる命令も書換え、間接番地指定の機能をもつ。なお、ノイマン型コンピュータはプログラム・データを蓄えるメモリ(手続き型言語モデルでは両者を分割して考えることがある)とCPU(さらに、バスを加えることがある)から成る。

### 演習問題

[4.1]  $0^n 1^n$  ( $n = 0, 1, 2, \dots$ ) を受理するチューリングマシンのアルゴリズムを示せ。

[4.2] 次の数論的関数  $f$  は原始帰納的関数であることを示せ。

$$(1) f(x, y) = x + y \quad (2) f(x) = x!$$

### 参考文献

1. 廣瀬健, 計算論, 近代数学講座 19, 朝倉書店, 昭 50 年.
2. 赤堀也, 伊藤正夫, 後藤英一, 広瀬健編, 情報処理のための数学, 共立出版, 昭 50 年.
3. 宮川洋, 情報論 —情報処理の理論—, 岩波全書, 1979 年.
4. 五十嵐善英, アルゴリズムと計算可能性, ソフトウェア講座 32, 昭晃堂, 昭 62 年.
5. L. ゴールドシュレーガー, A. リスター (武市正人, 小川貴英, 角田博保共訳), 計算機科学入門, 近代科学社, 1987 年.
6. 米田政明, 計算機科学の基礎, 森北出版株式会社, 1991 年.

---

<sup>5</sup>したがって、プログラムは自分自身を書換えることはできない。



# 5 アルゴリズムと計算量

アルゴリズムは与えられた問題を解く手順 (一般には仕事, すなわち処理をどのように行うかを記述したもの) である<sup>1</sup>. 当然コンピュータで問題を解くためには (コンピュータに向けた) アルゴリズムが存在しなければならない (計算可能性)<sup>2</sup>. アルゴリズムが存在しない問題もある. むしろ, アルゴリズムが存在しない (わからないのではない) 問題の方が存在する問題よりはるかに多い<sup>3</sup>のである. Turing は第 4 章で述べたように, チューリングマシンという仮想的な機械を用いて単純な命令の列としてアルゴリズムを記述した<sup>4</sup>. また, Church はラムダ計算という形式化を行い, Gödel は一組の規則の並びとしてアルゴリズムを記述している.

ここでは, アルゴリズムが存在する問題 (可解な決定問題) とはチューリングマシンで計算可能<sup>5</sup>な問題であると定義する<sup>6</sup>. また, 4.2 節で述べたように, もしチューリングマシンで実行可能なアルゴリズムが存在すればチューリングマシン, 抽象的計算機モデル, 現実のコンピュータへと (直観的にはインタプリタ (シミュレータ) を用いて) 計算可能性の同値性が成り立つ. このような考え方を **Church–Turing の提唱** という.

一方, 計算可能な問題はアルゴリズムが存在する, すなわちいつかは計算が終わり解が求まる (原理的に解ける) ということであって, 計算にかかる時間・必要な記憶容量については問わない. しかし, アルゴリズムが存在しても, 問題の規模が大きくなると実際に計算が不可能になっては意味がない. **計算量理論**では問題に要求される計算量 (時間, 記憶容量) について考える.

## 5.1 アルゴリズムの定義

チューリングマシン  $T_m$  によって計算可能な (すなわち, アルゴリズムが存在する) 問題を対象にする. そのために, まずアルゴリズムを次のように定義

---

<sup>1</sup>実際にはそれにデータ表現が加わる.

<sup>2</sup>どのようにアルゴリズムを設計するかという問題が**アルゴリズム論**である.

<sup>3</sup>このような議論は, コンピュータが出現する以前, 1930 年代に盛んに行われている. D.Hilbert はすべての数学的問題は, (1) 解答を出力する, (2) 解答を出力することが不可能であることを証明する, のいずれかであると考えた. そのため, 形式的体系の中で記述された問題の真偽を決定する (**決定問題**) アルゴリズムを考えたら, 実はそのアルゴリズムが存在しない (**決定不能**) のものがあることが証明された (**Gödel の不完全性定理**).

<sup>4</sup>計算論には, 第 4 章で述べたような**計算可能性理論**と, アルゴリズムを明示しこれにかかる計算量を求める**計算量理論**がある.

<sup>5</sup>すべての入力に対し問題を計算して停止するチューリングマシンが存在すること.

<sup>6</sup>Church の提唱では, アルゴリズムをもつ関数とは帰納的関数 (定義 4.3 参照. (1),(2),(3) の 3 つの関数を公理とし, (4),(5),(6) の 3 つの方法で定義される関数) のことである.

する.

**[定義 5.1] (アルゴリズム)** 問題を  $P$  とする. アルゴリズムとはその問題  $P$  を解く手順  $A$  のことで, 次の条件を満たす<sup>7</sup>.

- (1) (有限性) 有限個の操作 (または演算) から成る列であること.
- (2) (決定性) 1つの操作が実行されると次に実行すべき操作が決まっていること.
- (3) (終了性) 有限回の操作のあと必ず手順が終了すること<sup>8</sup>.

ここで, 問題  $P$  を適当に定義された入力集合  $X$  から出力集合  $Y$  への写像と考えれば, 問題を解くアルゴリズム  $A$  とはある入力  $x$  と出力  $y$  の対に対し

$$A : x \in X \rightarrow y = A(x) \in Y$$

と表現できる<sup>9</sup>(図 5.1 参照).

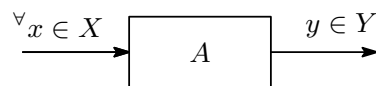


図 5.1: 問題  $P$  とアルゴリズム  $A$  の定義

なお,  $A : X \rightarrow Y$  が部分写像のとき,  $A$  をプロシージャ (手続き) とよんでアルゴリズムと区別することがある. すなわち, プロシージャは

- (i)  $A(x)$  が定義されているとき,  $y \in Y$  を出力して停止する
- (ii)  $A(x)$  が未定義のとき,  $y = \phi$  を出力して停止, または停止しないである.

## 5.2 アルゴリズムの構成法

ここでは, 個々の分野におけるアルゴリズム (例えばソート, マージなど) を紹介することが目的ではないので, アルゴリズムの設計手法の基本的・共通的・代表的な技法を述べる. 最適解を求めることが前提であるが, 問題の規模により準最適解を求める近似解法も実用的には重要である.

<sup>7</sup>もちろん, アルゴリズムは唯一ではない.

<sup>8</sup>これに加え (4) (解の導出性) 手順が終了したとき, 問題  $P$  の答  $y$  が得られていること, を加えることがある.

<sup>9</sup>ここで, 操作とはチューリングマシン  $T_m$  の基本命令に相当する. すなわち, テープの読み・書き込み操作と命令実行に対応する有限個の状態間の遷移操作 (これがプログラムに対応する) である. これでは余りにめんどうなので, 一般には RAM を仮定し, 演算と接続・分岐・繰返しなどの制御を基本操作とする.

**(1) 分割統治法 (divide-and-conquer 法)**

問題  $P$  をいくつかの小規模な部分問題  $P_1, P_2, \dots, P_k$  に分割し、それらの解を結合して全体の解を得る方法である。分割は対象とする変数集合あるいは変数の定義域などに対して行う。この結果、アルゴリズムが再帰的に反復して適用される。

例えば、2分探索、クイックソート、マージソート、高速フーリエ変換 (FFT)、行列の積<sup>10</sup>などがある。

**(2) 動的計画法 (dynamic programming)**

最適化問題で解がいくつかの決定結果から得られる系列値の中の最大 (または最小) 値で定まるとする。決定過程を状態間の遷移としてとらえ、状態をまとめることができるとき、対象となる部分問題の解を計算・記憶しこれを逐次用いることにより最適性を保存したまま計算の手間を減少させることができる。もともと制御理論で開発された手法で、次に述べる**最適性原理**に根拠をおいている。典型的な適用例にマルコフ決定過程がある<sup>11</sup>。

**[定理 5.1] (最適性原理)** 最適な決定の列  $(d_1, d_2, \dots, d_N)$  では初期状態と最初の決定  $d_1$  がどんなものであっても残りの決定の列  $(d_2, d_3, \dots, d_N)$  は  $d_1$  の結果生ずる状態を初期状態とする最適な決定の列である。 □

これは任意の決定  $d_i$  に対しても成り立つ。ネットワークの最適パス (コスト最小のパスを求める問題。例えば CPM (Critical Path Method)) に適用すれば、次のように述べることができる。初期ノード  $S$  から最終ノード  $T$  へ至るあらゆるパスを考える。その間の2本がノード  $M$  で合流するとき、もし  $S - r_1 - M - T$  が最適であるとすれば、 $S$  から  $T$  に至るあらゆるパスの中で  $S - r_1 - M$  が最適でなければならない。したがって、 $S - r_2 - M - T$  は最適パスになり得ないから、 $M$  において  $S - r_1 - M$  だけを残し  $S - r_2 - M$  を捨て

<sup>10</sup>シュトラッセンの方法を用いると  $n \times n$  の行列  $A, B$  の積の計算量は  $O(n^{\log 7}) = O(n^{2.81})$  だよ。通常の行列を分割しないと明らかに  $O(n^3)$  である。ただし、シュトラッセンの方法は定数係数が大きく実用的ではない。なお、単純な  $n$  桁の2整数  $A, B$  の乗算  $A \times B$  も通常  $O(n^2)$  を必要とするが  $A = (A_1, A_2), B = (B_1, B_2)$  と分割し  $A_1 B_1, (A_1 + A_2)(B_1 + B_2) - A_1 B_1 - A_2 B_2, A_2 B_2$  を計算し桁シフトして加えれば  $O(n^{1.59})$  で計算できる。現在、最も早い方法では  $O(n \log n \log \log n)$  で計算できることが知られている。

<sup>11</sup>動的計画法における部分問題は分割統治法における部分問題とは本質的に異なる。後者では問題を分割してできる部分問題は個々に計算して解を求める。したがって、部分問題がさらに小さな部分問題に分割されたとき同一の問題が現れても、それぞれ別個に計算しなければならない。これに対し前者は一度計算して得た解は記憶しておき再び用いることができる。

ることができる。

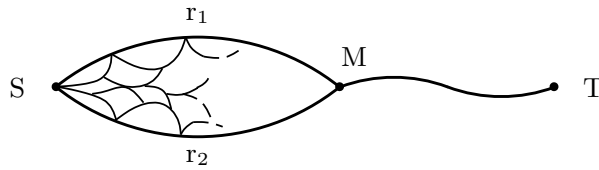


図 5.2: 最適性原理の説明図

この原理を再帰的に用いることにより、漸化式を求め最適解を得る方法が動的計画法である。

例えば、最適 2 分探索問題、最短路探索問題 (CPM など)、ナップザック問題などの他、ビテルビ (Viterbi)<sup>12</sup> 復号法などの適用がある。

### (3) 分枝限定法 (branch-and-bound method)

効率の良い解法が知られていない問題も少なくない。そのような場合、解の候補となるすべてを列挙し片っぴしから探索する方法 (しらみつぶし法 (exhaustive search)) がとられる。これを列挙法 (enumeration method) という。しかし、大規模問題では実際に適用するのが難しい。

そこで、まず問題をいくつかの部分問題に分割する (分枝操作)。この分割は通常、解空間を木構造で分枝させる (例えば、ある変数  $x_i$  が  $x_i \geq 0$  と  $x_i < 0$  の 2 つの解空間に分枝させる。分割統治法の部分問題の分解が変数集合を分割するのは異なる)。次に、分枝によって次第に小規模となった部分問題の最適解が求まる場合、あるいは何らかの方法で最適解になり得ないことが示せる場合、これらを打ち切る (限定操作、枝切り操作)。すでに解かれた部分問題の最適解を保持し、解くべき部分問題がなくなったとき、保持されている最適解を出力する。

最適解にはなり得ない部分問題の抽出法 (限定法) には、例えば最小となる最適解を求める問題で

- (1) すでに解いた部分問題  $P_i$  の最適解が  $y_{i0}$  (下界値) のとき、部分問題  $P_j$  の最適解  $y_{j0} > y_{i0}$  ならば  $P_j$  を限定してよい。
- (2)  $P_i$  の許容解が  $P_j$  の許容解を含む ( $P_i$  の許容解が  $P_j$  の許容解より常に大きい) とき、 $P_i$  を限定してよい。

などである。なお、人工知能の分野で開発された A\* アルゴリズムは本質的には分枝限定法と等価である (後述)。

<sup>12</sup>ピタビと読むことも多い。



例えば、巡回セールスマン問題、0-1 ナップザック問題など組合せ最適化問題に多くの適用例がある。

#### (4) その他の解法

列挙法 (全探索法) などのアルゴリズムを工夫しても実用的な限界がある。現実には最適解を求めるのが不可能な場合、近似的にでも解 (準最適解, 近似最適解) を見つけ出さねばならない問題も数多くある。ここでは、そのような近似解法について述べる。

- (a) **局所探索法** (local search method) 可能解 (初期解) を1つ選び、これを暫定解としてその近傍を探索する。暫定解より良い解が見つければこれをあらためて暫定解とし再びその近傍を探索する。これを繰返し暫定解より良い解が見つからなくなれば停止する。局所最適解は全域最適解となっている保証はないが、初期解を列挙法またはランダムに変更して局所探索を繰り返す方法がとられる。

1変数ずつ評価関数 (目的関数) に対する局所的評価値の影響の大きな変数を順次加えてゆく**グリーディ法** (greedy method, 欲張り法ともいう)<sup>13</sup>、逆に、すべての変数に対する評価値を求め局所的な評価から不要な変数を除去する**スティンジー法** (stingy method, けちけち法ともいう) がある。

また、局所探索法の解の変更方法を金属の焼きなましを模擬した**シミュレーテッドアニーリング法** (simulated annealing method)、生物の進化過程を模擬した**遺伝的アルゴリズム** (genetic algorithm)、局所解を避けるために以前の実行結果を記憶し、不良なものを除外条件とする**タブーサーチ法** (tabu search method) などのメタ戦略と呼ばれる手法がある。

- (b) **発見的探索法** (heuristic search method) 探索の過程で今後の評価値を予測できれば効率良い探索ができる。予測を行うための知識を発見的 (heuristic) に見出し、これを探索に用いる。(a) で述べた局所探索法も多くの場合、全探索ではなく少しでも良い方向に向かう発見的知識が用いられているから発見的探索法に分類することができる。

探索グラフの各節点  $m$  において出発点  $S$  からの最適コストを  $g(m)$ 、 $m$  から目標点  $T$  までの真のコストを  $h(m)$  とする。何らかの方法で  $h(m)$  に対しその予測値  $\hat{h}(m)$  が求まれば節点  $m$  を通る全コスト  $\hat{f}(m) = g(m) + \hat{h}(m)$  を用いた枝切りのアルゴリズムを構成することができる。ただし、 $\hat{h}(m)$  が  $h(m)$  の

<sup>13</sup>最短経路を求めるダイクストラアルゴリズムはグリーディ法に基づいたもので、常に最適解が得られる。

良い予測値になっていれば探索効率は良くなるが、良い予測値でない場合はむしろ効率が悪くなり、最適解が見つからないこともある。

このような方法に**最良優先探索** (best-first-search), **A アルゴリズム**, **A\* アルゴリズム**がある。全節点  $m$  において,  $\hat{h}(m) \leq h(m)$  が成り立つとき, **A\* アルゴリズム**といい, 最適解が得られる保証がある<sup>14</sup>。

この他, 近似解法には制約条件の一部を緩和 (例えば, 0-1 計画法を実数領域で解く) し, 得られた解を可能解に修正する**緩和法** (relaxation method) がある。

### 5.3 計算量の評価

#### (1) 計算量の定義

計算量 (computational complexity) はアルゴリズムの複雑さを測る尺度である。あるアルゴリズムに入力  $x$  を投入し, 出力  $y$  を得るまでに費やした計算時間を示す**時間計算量** (time complexity) と使用した記憶領域の量を示す**領域計算量** (space complexity) がある。

いま, 問題の規模 (大きさ) を示すパラメータを  $n$  とする。  $n$  を大きくしていくときの漸近的な (asymptotic) 時間計算量を**漸近的時間計算量**, また漸近的な領域計算量を**漸近的領域計算量**という。コンピュータの資源が限定されたとき, 与えられたアルゴリズムで解くことができる最大の問題の大きさは漸近的計算量によってきまる。漸近的計算量は, 通常次の **Landau の記号**を用いて示される。すなわち

$$\forall n, \exists c(\text{定数}) \quad |f(n)| \leq c|g(n)| \quad \text{ならば} \quad f(n) = O(g(n)) \quad (5.1)$$

と表す<sup>15</sup>。

さらに, 最悪の計算量あるいは平均計算量を問題としなければならないときも多い。いずれも, 時間計算量, 領域計算量に対し**最悪 (最大) 計算量** (worst-case complexity), **平均計算量** (average-case complexity) が定義される。

<sup>14</sup>分枝限定法を探索グラフに適用したとき, 節点  $m$  を通る最適コスト  $f(m)$  の代わりに  $\hat{f}(m) \leq f(m)$  を満足する下界値  $\hat{f}(m)$  を用いて枝切り (限定操作) を行うアルゴリズムと考えることができる。  $\hat{f}(m)$  がそれまでに得られている暫定最適コストより大きければ節点  $m$  の小節点は枝切り (限定) してよい。すなわち, ある  $S$  から  $T$  のパスを初期解とし, これを暫定最適解としてこれより良いパスを探索する方法である。 **A\*** アルゴリズムは  $\hat{h}(m)$  を用いて枝切りを行うが本質的に両者は等価である。

<sup>15</sup>“ $f(n)$  はオーダー  $g(n)$  である”と読む。

これらを定義するには

- (1) 問題の大きさを示すパラメータ  $n$
- (2) 入力  $x$  を与えて出力  $y$  を求める計算機械モデル

の2両者が明確に与えられねばならない。これらはいずれも、次に示すように、4.1節で述べたチューリングマシンという最も単純で原始的な機械を用いて定式化される。

## (2) 問題の記述

アルゴリズムは入力テープに記述された問題の語 (入力データ) をチューリングマシン  $T_m$  が受理する問題として定式化される。すなわち、定義 4.1 でテープの入力記号列  $x$  を

$$x = x_1x_2 \cdots x_n, \quad x_i \in T \quad (x \in T^*) \quad (5.2)$$

とする。テープ記号の有限集合  $T$  により長さ  $n$  の記号列として記述し<sup>16</sup>、 $T_m$  がこれを受理したとき、 $x$  の系列長 (入力サイズ)  $n$  が問題の記述長を表し、これを問題の規模を示すパラメータとする。しかし、これでは問題の記述方法 (符号化)、したがってアルゴリズムの構成法によっても変わり、一般に余りにも煩雑である。通常はパラメータ  $n$  の値として、例えばソーティングの場合は入力データの個数、正方行列の積を求める場合は行 (あるいは列) の個数 (次数)、連立方程式の場合は変数の個数 (元の数) などが用いられる。

## (3) 計算機械モデル

アルゴリズムの計算量は実行する計算機械のステップ数 (時間計算量) と使用記憶容量 (領域計算量) で量る。ここでは、抽象的計算機械モデルとして通常、チューリングマシン  $T_m$  (実際には  $k$  テープ  $T_m$  を考えればわかりやすい) を用いる。

入力  $x$  が与えられたとき、( $k$  テープ)  $T_m$  がこの入力を受理するとき、すなわち  $T_m$  が  $x$  を受理しいつか解  $y$  を出力して止まるとき<sup>17</sup>、それまでに要したステップ数 (状態遷移回数、あるいはヘッドの移動回数) がたかだか  $T(n)$  のとき、 $T_m$  の時間計算量は  $T(n)$ 、また使用したテープのコマの最大数がたかだか  $S(n)$  のとき、 $T_m$  の領域計算量を  $S(n)$  という。

チューリングマシン  $T_m$  の拡張として  $k$  テープ  $T_m$  の他に万能チューリングマシン  $U$  がある。また、コンピュータに近い RAM や RASP (4.2 節参照) が

<sup>16</sup>\* は任意の系列長を示す。

<sup>17</sup> $x$  を受理しないで止まる場合と、いつまでも動き続けるときは受理しないという。

ある。さらに、具体的コンピュータとなると多種多様で、使用言語（マシン語、アセンブラ言語、高級言語など）によりかなり異なる。しかし、幸いこれらの間には適当な条件<sup>18</sup>の下に、ステップ数の換算（直観的にはインタプリタの処理ステップ数の比）が可能である。その結果、時間計算量・領域計算量に対して、たとえば次のことが知られている。

- (i) 任意の  $T_m$  は互いに定数係数の範囲内で同等である<sup>19</sup>。
- (ii) RAM と RASP は定数係数の範囲内で同等である。

一方、 $DT_m$ （決定性  $T_m$ ）と RAM の間には模倣可能という考え方をを用いて次のように換算する。時間計算量が  $T(n)$  の  $T_m$  に対し、同じ入力  $x$  を受理する RAM は

- (i) RAM は  $DT_m$  で  $O(n^3)$  時間模倣可能
- (ii)  $DT_m$  は RAM で  $O(n^2)$  時間模倣可能

であることが知られている。ただし、RAM の各命令は 1 単位時間で実行され、どのレジスタも 1 単位領域を占有する（一様コスト基準）ものとする。

以上のような手順により、本質的な評価を損なうことなく異なった計算機で計算量を示すことができる。そのため、計算時間を左右する本質的な演算回数により評価することも行われている<sup>20</sup>。なお、漸近的な計算量の評価においては、定数倍が影響しないことも評価を簡潔にさせている。例えば、 $O(n^2) + O(n^2) = O(n^2)$ ,  $O(n^3) + O(n^2) + O(\sqrt{n}) = O(n^3)$  である。

#### 5.4 問題の複雑さのクラス

チューリングマシン  $T_m$  を用いて問題を計算量によって分類する。

##### (1) 計算量の階層

決定性 (D)、非決定性 (N) のチューリングマシン  $DT_m$ ,  $NT_m$  を用いて時間計算量が  $O(T(n))$  のクラスをそれぞれ D Time( $T(n)$ ), N Time( $T(n)$ ), 同様に領域計算量が  $O(S(n))$  のクラスをそれぞれ D Space( $S(n)$ ), N Spase( $S(n)$ ) と表す。このとき、以下のことが知られている。

決定性と非決定性による計算量の違いとして

<sup>18</sup>例えば、RAM, RASP の命令には乗算はない。

<sup>19</sup>領域計算量（テープ圧縮定理）、時間計算量（線形加速定理）に対し、それぞれ定数倍にできる。

<sup>20</sup>基本演算の回数であって実行時間（CPU タイムなど）ではない。実際の実行時間はプログラム言語、プログラムの書き方によっても大きく変わるから、アルゴリズムの計算量の評価の目安になっても普遍的なものではない。

(i)  $\text{N Time}(T(n)) \subseteq \bigcup_c \text{D Time}(c^{T(n)})$ . ただし,  $c$  は定数

(ii)  $\text{N Space}(S(n)) \subseteq \overset{c}{\text{D Space}}(S^2(n))$

非決定性領域計算量について

(iii)  $\text{N Space}(n^r) \subseteq \text{N Space}(n^{r+\varepsilon})$  ( $\varepsilon > 0, r \geq 0$ )

領域計算量と時間計算量について,  $f(n) \geq n, g(n) \geq \log n$  とするとき

(iv)  $\text{D Time}(f(n)) \subseteq \text{D Space}(f(n))$

(v)  $\text{D Space}(g(n)) \subseteq \bigcup_c \text{D Time}(c^{g(n)})$

である.

## (2) P と NP

まず, 重要な問題のクラスを定義する. 入力の大きさを  $n$  とするとき

$\mathcal{DL}$  =  $\text{D Space}(\log n)$  : 決定性対数領域計算可能な問題のクラス

$\mathcal{NL}$  =  $\text{N Space}(\log n)$  : 非決定性対数領域計算可能な問題のクラス

$\mathcal{P}$  =  $\bigcup_i \text{D Time}(n^i)$  : 決定性多項式時間計算可能な問題のクラス

$\mathcal{NP}$  =  $\bigcup_i \text{N Time}(n^i)$  : 非決定性多項式時間計算可能な問題のクラス

$\mathcal{P Space}$  =  $\bigcup_i \text{D Space}(n^i)$

=  $\bigcup_i \text{N Space}(n^i)$  : 多項式領域計算可能な問題のクラス

$\mathcal{EXP Time}$  =  $\bigcup_i \text{D Time}(2^{n^i})$  : 決定性指数時間計算可能な問題のクラス

とする. これらの間に次式が成り立つことが知られている.

$$\mathcal{DL} \subseteq \mathcal{NL} \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{P Space} \subseteq \mathcal{EXP Time} \quad (5.3)$$

ただし, 狭義の包含関係としては

$$\mathcal{DL} \subsetneq \mathcal{P Space}, \quad \mathcal{P} \subsetneq \mathcal{EXP Time} \quad (5.4)$$

が知られているのみである.  $\mathcal{P} \subseteq \mathcal{NP}$  は示されているが,  $\mathcal{P} = \mathcal{NP}$  が成り立つかどうか (P・NP 問題) は未解決である<sup>21</sup>.

## (3) NP 完全問題

<sup>21</sup> $\mathcal{NP}$  は例えば組合せ問題で, 列挙法を同時に計算できるような計算機械で多項式時間計算可能な問題のクラスである. したがって, 並列コンピュータのモデルに相当する (前述). ただし, 組合せの数が爆発してもよいとしているからプロセッサの数は無限に必要である.  $\mathcal{P} \subsetneq \mathcal{NP}$  も明らかのようにみえるが証明されていない.

[定義 5.2] (還元可能 (reducible)) 言語  $L_1$  が言語  $L_2$  に還元可能であるとは、計算量  $f(n)$  のある D Tm が存在して次式が成り立つことである<sup>22</sup>.

$$\forall x \in L_1 \iff y = A(x) \in L_2 \quad (5.5)$$

□

ここで、 $f(n)$  が多項式時間のとき、 $L_1$  は  $L_2$  に多項式時間還元可能といい  $L_1 \leq_p L_2$  と書く。  $f(n)$  が対数領域のとき、 $L_1$  は  $L_2$  に対数領域還元可能といい  $L_1 \leq_{\log} L_2$  と書く。

[定義 5.3] (困難 (hard) と完全 (complete))  $\mathcal{C}$  を任意の計算量のクラスとする。

- (1)  $L_1 \in \mathcal{C}$ ,  $L_1 \leq_p L_2$ , または  $L_1 \leq_{\log} L_2$  のとき、 $L_2$  は  $\mathcal{C}$  困難
- (2) (1) を満足し、かつ  $L_2 \in \mathcal{C}$  のとき、 $\mathcal{C}$  完全という。

□

問題のクラス  $\mathcal{C}$  を  $\mathcal{NP}$  とすると、問題  $A_1 \in \mathcal{NP}$  が  $A_2$  に多項式還元可能であれば  $A_2$  は NP 困難であり、さらに  $A_2 \in \mathcal{NP}$  ならば  $A_2$  は NP 完全である。すなわち、NP 困難とは  $\mathcal{NP}$  に属するどの問題と比較してもそれ以上難しいこと、NP 完全とは  $\mathcal{NP}$  の中で最も難しい問題である。したがって、ある問題が NP 完全問題であることを示せば<sup>23</sup>実際的には多項式時間では解けないことを示唆 (予想) していることになる。すなわち、 $P \subsetneq \mathcal{NP}$  は未解決であるが  $\mathcal{NP} \setminus P$  に属する問題があれば NP の中で最も難しい問題である。一般に、実際的な時間内で解けるという意味で実際的計算可能なクラスは  $P$  が妥当であると考えられている。逆に、 $P$  に属さない問題は実際的でないと考えてよい。

NP 完全問題にはハミルトン閉路問題、巡回セールスマン問題、3色可能問題など数多くあり、いずれも多項式時間アルゴリズムは見つかっていない。

## 演習問題

### [5.1] ユークリッドの互除法について

<sup>22</sup> $m(x)$  を計算するアルゴリズムが存在するとする。問題  $A_1, A_2$  を  $L_1, L_2$  を受理する Tm と考えると、 $A_1$  の入力  $\forall x$  に対し  $A_2$  の入力  $y = m(x)$  を構成でき、 $A_1$  における  $x$  の答 {Yes, No} と  $A_2$  における  $y = m(x)$  の答が一致するとき、 $A_1$  は  $A_2$  に還元可能 (帰着可能ともいう) である。

<sup>23</sup>そのためには、与えられた問題  $A$  がある NP 完全問題を  $A$  に多項式還元し、さらに  $A \in \mathcal{NP}$  であることを示せばよい。

- (1) (色んな方法で) アルゴリズムを記述せよ.
- (2) PASCAL を用いてプログラムを示せ.
- (3) 計算量を求めよ.

[5.2] ある NP 困難問題  $A$  が,  $A \in \mathcal{P}$  ならば  $\mathcal{P} = \mathcal{NP}$  が導けることを示せ.

## 参考文献

1. D.E.Knuth, The art of computer programming Vol3/Sorting and Searching, Addison-Wesley Publishing Company, 1973 年.
2. 赤堀也, 伊藤正夫, 後藤英一, 広瀬健編, 情報処理のための数学, 共立出版, 昭 50 年.
3. N ヴィルト, (浦昭二, 國分方久史共訳), アルゴリズムとデータ構造, 近代科学社, 1986 年.
4. 五十嵐善英, アルゴリズムと計算可能性, ソフトウェア講座 32, 昭晃堂, 昭 62 年.
5. L. ゴールドシュレーガー, A. リスター (武市正人, 小川貴英, 角田博保共訳), 計算機科学入門, 近代科学社, 1987 年.
6. 野崎昭弘, アルゴリズムと計算量, 計算機科学/ソフトウェア技術講座 5, 共立出版, 1987 年.
7. R. セジウィック, (野下浩平, 星守, 佐藤創, 田口東共訳), アルゴリズム, 第 1 巻=基礎整理, 近代科学社, 1988 年.
8. R. セジウィック, (野下浩平, 星守, 佐藤創, 田口東共訳), アルゴリズム, 第 2 巻=探索文字列計算幾何, 近代科学社, 1988 年.
9. 茨城俊秀, アルゴリズムとデータ構造, 昭晃堂, 1989 年.
10. 有澤誠, アルゴリズムとその解析, コロナ社, 1989 年.
11. 足立暁生, アルゴリズムと計算理論, 情報工学入門シリーズ 6, 森北出版株式会社, 1990 年.
12. 米田政明, 計算機科学の基礎, 森北出版株式会社, 1991 年.
13. 川合慧, コンピューティング科学, 東京大学出版会, 1995 年.
14. 斎藤信男, 西原清一, データ構造とアルゴリズム, コロナ社, 平成 10 年.





## 6 プログラム理論

我々は日頃、自分の作ったプログラムがなかなか思ったように動作してくれないことを経験する。新しく開発したアプリケーションプログラムをリリースすると必ずといってよい程クレームがありプログラム修正を行う。流通しているソフトウェア(プログラム)もメンテナンスと称してバージョンアップ(多少ともプログラム修正が含まれる)が図られる。これらは、いずれも人間の手によるだけでは大規模なプログラムを短期間に誤り(バグ)なく作成することはほとんど不可能であることを示している。このようなバグは、昔アメリカが金星に向かって発射した探査ロケットの制御プログラム<sup>1</sup>による誤動作、最近では鉄道が止ったり、バンキングシステムの停止などのように多大な損害をもたらす。プログラム理論はこのようなプログラムのバグをコンピュータによって見つけ出すために考えられたものである。すなわち、プログラムの意味や正しさの検証を形式的に行うための数学的理論が**プログラム理論**である。

そのためにプログラムの意味を厳密に与える**形式的意味論**(formal semantics)と、これを用いてその正しさを証明する**プログラムの証明論**(検証論)(program verification)が必要となる。したがって、要求定義からこれを満たすプログラムを生成する**自動プログラミング**(プログラムの自動生成・合成)と密接な関係にある。

プログラムのバグには、①**構文誤り**(syntax error)、②**意味誤り**(semantic error)、③**実行(時)誤り**(execution error)がある。①は主としてプログラム言語の文法に対する構文誤りであり、例えば、コンパイル時にシンタックスエラーあるいはワーニングとして出力される。③は2000年問題のように前提条件が変わった場合や定義されていない入力データなどに対して異常動作をする誤りである。ここでは、②に対する検証が中心である。②は実行前に(未然に)誤りを防ぐことが可能であることに注意する。

**プログラム理論**は1967年、R.W.Floydにより研究が開始され、1969年にはJ.C.Kingによりプログラム検証ツールが開発されている。しかし、その後提案・開発された多くの手法は実務用としてはほとんど使用されていない。その理由は本質的な理論限界<sup>2</sup>の他、前提条件と効果を厳密に記述することはほとんどプログラムを書くことに等しく約2倍のコストがかかるため、事務処理など定型的なものを除いて実用化が進んでいない<sup>3</sup>。ただし、手続き型言語の部品合成による自動プログラミングはいくつかの事例がある。

---

<sup>1</sup>“,”(カンマ)とすべきところを“.”(ピリオド)となっていた。そのため、繰り返し文が代入文となっていたという有名な話。

<sup>2</sup>完全性が成り立たない(自然数の理論を形式化して得られる理論体系では理論式  $A$  とその否定  $\neg A$  が共に証明不可能な理論式  $A$  が存在する)。

<sup>3</sup>実際、要求定義の形式的仕様記述作業とプログラム開発作業はほとんど等しい。

### 6.1 プログラムの意味論

プログラムの意味を表現するにはいくつかの方法がある。プログラムをどう定義するかにより次のように分けられる。

#### (a) 関数とみる立場

- ① 表示の意味論 … プログラムをグラフ理論における関数方程式と考え、その最小不動点をプログラムの意味とみなす方法
- ② 操作的意味論 … プログラムの抽象的機械 (オートマトンなど) 上の動作によって意味を記述する方法

#### (b) 論理とみる立場

- ③ 公理の意味論 … プログラム言語の意味を推論システムの公理と推論規則により与えた論理体系に基づきプログラムの意味を記述する方法

ここでは、③について C.A.R.Hoare によるプログラムを定理とみなす立場 (**Hoare 理論**) を簡単に述べる。

手続き型言語によるプログラム  $P$  を考える。  $\varphi, \Psi$  を述語理論式とし、“状態  $\varphi$  でプログラム  $P$  を実行し停止すれば、状態  $\Psi$  を満たす” を

$$\varphi\{P\}\Psi \quad (6.1)$$

と書く。これを**表明**という。  $\varphi, \Psi$  は変数の値に関する記述であり、  $P$  はステートメント (例えば代入文) である。したがって、  $P$  の意味は変数の値の変換操作である。例えば、  $x_i$  を  $a_i (i = 1, 2, \dots, n)$  でおきかえる代入文は

$$\varphi\{(x_1, x_2, \dots, x_n) := (a_1, a_2, \dots, a_n)\}\Psi \quad (6.2)$$

で記述される。

### 6.2 プログラムの証明論

公理的意味論を用いた Hoare 理論では、プログラムの構造にしたがった公理化を行う。プログラム  $P$  が用いるプログラム言語の文法 (構文則) にしたがって記述されていることから、部分的正当性を示すために構文要素に対応する検証 (推論) 規則を次のように与える。

#### ① 代入規則

$$\frac{A \supset B_{a_i}^{x_i}}{A\{(x_1, x_2, \dots, x_n) := (a_1, a_2, \dots, a_n)\}B} \quad (6.3)$$

② if 規則

$$\frac{(A \wedge P)\{R_1\}B, (A \wedge \neg P)\{R_2\}B}{A\{\text{if } P \text{ then } R_1 \text{ else } R_2\}B} \quad (6.4)$$

③ while 規則

$$\frac{A \supset I, (I \wedge P)\{R\}I, I \wedge \neg P \supset B}{A\{\text{while } P \text{ do } R\}B} \quad (6.5)$$

④ 接続規則

$$\frac{A\{P_1\}B, B\{P_2\}C}{A\{P_1, P_2\}C} \quad (6.6)$$

ここで、式 (2.6.3)–(2.6.6) の横棒の下部の式を推論するためには上部の式を証明すればよいことを示す。また、 $B_{a_i}^{x_i}$  は術語  $B$  に現れるすべての  $x_i = a_i$  ( $i = 1, 2, \dots, n$ ) として得られる術語を示す。

検証は論理学のシンタックスを用いて検証単位の論理式  $\varphi\{P\}\Psi$  が定理になっていることを証明することにより実行される。

## 参考文献

1. 横内寛文, プログラム意味論, 共立出版株式会社, 1994 年.



# 7

## 情報理論

情報理論は1948年、C.E.Shannonの論文“A mathematical theory of communication”により誕生した。広義にはN.Wienerによるサイバネティックス (cybernetics), R.E.Kalmanによる予測理論, H.Nyquistらによる伝送理論や信号解析・雑音理論なども含まれる。ここでは、Shannon流の情報理論から3つの重要なテーマである情報量の定義・情報源符号化・通信路符号化について述べる。

情報の確率的構造に注目した情報量の定義は多種多様な情報を定量的に取り扱うことができる尺度として価値があるばかりではなく、集合論・関数解析・確率統計などの数学を有効に利用し、すっきりした理論体系を作り上げることを可能にしている。また、符号化(ベクトルの写像)という操作により情報の中味を損うことなく冗長度を除去(データ圧縮)したり、逆に誤りが生起しても情報の中味を正しく復元するために冗長度を付加(誤り訂正)することができる。

なお、マルチメディアの**メディア**とは情報の物理的表現形態を指し、文字・音声・図形・静止画像・動画像など多種多様な多元的メディアを扱う。7.2節で述べる標本化・量子化という操作は音声や画像など、どんなアナログ情報もある適当な条件の下にすべてデジタル情報に変換可能であることを示している。したがって、コンテンツのデジタル情報化によりマルチメディアを一元的に符号化することが可能となる<sup>1</sup>。情報理論はこのようなマルチメディアの情報基盤の構築に寄与しているのである。

### 7.1 情報量の定義

#### (1) 自己情報量

---

<sup>1</sup>その結果、マルチメディア PC など一台のコンピュータですべて処理(計算・記憶(蓄積)・通信)が可能となる。

**[定義 7.1] (自己情報量)** ある事象  $a$  が起こる確率を  $\Pr(a) = p_a$  とする. 次の性質をもつ関数  $I(a)$  を  $a$  の実現に関する**自己情報量**と定義する.

- (1)  $I(a) \geq 0$ . ただし,  $I(a) = 0$  は  $p_a = 1$  の場合に限る.
- (2)  $I(a)$  は  $p_a$  の連続単調減少関数である.
- (3) 二つの独立事象  $a, b$  の実現する確率をそれぞれ  $p_a, p_b$  とする. このとき,  $a$  と  $b$  とが同時に起こる確率は  $p_a \cdot p_b$  であり

$$I(a \cap b) = I(a) + I(b) \quad (7.1)$$

が成り立つ. □

$I(a) = i(p_a)$  とおき, 上の 3 つの性質をもつ関数を求めると, 一般に

$$i(p) = -\alpha \log_e p \quad (\alpha > 0) \quad (7.2)$$

である.  $\alpha = 1/\log_e b$  とおくと

$$i(p) = -\log_b p \quad (b > 0) \quad (7.3)$$

と表わせる. 対数の底  $b$  により情報量の単位 (呼称) を定める.  $b = 2$  のとき [bit] または [shannon],  $b = e$  のとき [nat],  $b = 10$  のとき [Hartley] または [decit] などがある.

**[定理 7.1]** 定義 7.1 を満足する**自己情報量**は次式で与えられる.

$$I(a) = -\log_2 p_a \quad [\text{bits}] \quad (7.4)$$

## (2) エントロピー

$n$  個の異なった互いに排他的 ( $a_i \cap a_j = \emptyset (i \neq j)$ ) な事象  $a_i (i = 1, 2, \dots, n)$  があり, それぞれの事象  $a_i$  の起こる確率  $\Pr(a_i) = p_i (i = 1, 2, \dots, n)$  とする. すなわち, 情報源  $\mathcal{A}$  が完全事象系するとき

$$\mathcal{A} = \left[ \begin{array}{c} a_1, a_2, \dots, a_n \\ p_1, p_2, \dots, p_n \end{array} \right] \quad \left( \sum_{i=1}^n p_i = 1 \right) \quad (7.5)$$

で表す. 完全事象系  $\mathcal{A}$  に対して, その**エントロピー** (entropy) (または**平均情報**

量) を

$$H(\mathcal{A}) = - \sum_{i=1}^n p_i \log_2 p_i \quad [\text{bits/symbol}] \quad (7.6)$$

で与える.

**[例 7.1] (2元エントロピー関数)**

式 (7.6) で  $n = 2$  の場合を考える. これは情報源  $\mathcal{A}$  が  $\{0, 1\}$  からなり, 0 と 1 の生起確率をそれぞれ  $p, 1 - p$  とするとき, エントロピーは

$$H(\mathcal{A}) = -p \log_2 p - (1 - p) \log_2 (1 - p) \triangleq H_b(p) \quad (7.7)$$

となる. 関数  $H_b(p)$  を **2元エントロピー関数** という. 図 7.1 に示すように  $p = 1/2$  のとき  $H_b(p)$  は最大値をとり,  $H_b(1/2) = 1$  [bit] である. また,  $p = 1, p = 0$  は観測しなくても初めからそれぞれ 0 あるいは 1 であることがわかっており, 結果をみても新たに得る情報はなにもなく, エントロピー  $H_b(0) = H_b(1) = 0$  である.  $\square$

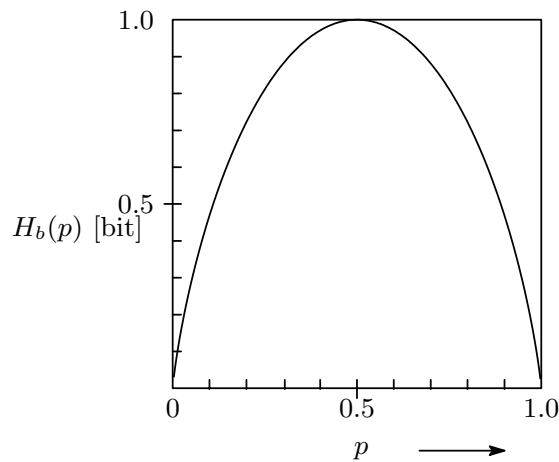


図 7.1: 2元エントロピー関数  $H_b(p)$  と確率  $p$  の関係

**(3) 相互情報量**

自己情報量  $I(a)$  を拡張してエントロピー  $H(\mathcal{A})$  を求めたと同様, 2つの完全事象系  $\mathcal{A}, \mathcal{B}$  から相互情報量  $I(\mathcal{A}; \mathcal{B})$  を導く. 事象系  $\mathcal{A}$  は式 (7.5) で与えられるものとし,  $q_i = \Pr(b_i)$  とおいて事象系  $\mathcal{B}$  を次式で与える.

$$\mathcal{B} = \begin{bmatrix} b_1 & b_2 & \cdots & b_m \\ q_1 & q_2 & \cdots & q_m \end{bmatrix} \quad \left( \sum_{i=1}^m q_i = 1 \right) \quad (7.8)$$

次式で与えられる  $I(\mathcal{A}; \mathcal{B})$  を**相互情報量**という.

$$I(\mathcal{A}; \mathcal{B}) = \sum_{j=1}^m \sum_{i=1}^n \Pr(a_i, b_j) \log \frac{\Pr(a_i, b_j)}{\Pr(a_i) \Pr(b_j)} \quad [\text{bits/symbol}] \quad (7.9)$$

ただし,  $\Pr(a_i, b_j)$  は事象  $a_i$  と事象  $b_j$  の同時確率で, 事象  $a_i$  の条件付の事象  $b_j$  の生起確率を  $\Pr(b_j|a_i) = P_{ij}$  とするとき

$$\Pr(a_i, b_j) = \Pr(a_i) \Pr(b_j|a_i) = p_i P_{ij} \quad (7.10)$$

である.

## 7.2 情報源符号化

### (1) 標本化と量子化

時間  $t$  の連続関数 (連続信号)  $f(t)$  を考える.  $f(t)$  がある周波数帯域  $[-f_0, f_0]$  [Hz] に制限されているとき, すなわち  $f(t)$  の周波数スペクトル  $F(\omega)$  が  $F(\omega) = 0$  ( $|\omega| > \omega_0, \omega_0 = 2\pi f_0$ ) であるとき, フーリエ級数を用いて  $F(\omega)$  を求めることができる. その逆変換を求めると, 整数  $m$  に対し

$$f(t) = \sum_{m=-\infty}^{\infty} f\left(\frac{m}{2f_0}\right) \frac{\sin \pi(2f_0 t - m)}{\pi(2f_0 t - m)} \quad (7.11)$$

と表わすことができることが知られている (演習問題 [7.3] 参照). その結果, 直ちに次の定理が得られる.

**[定理 7.2] (染谷・Shannon の標本化定理)** 連続信号  $f(t)$  の周波数帯域が有限な  $[-f_0, f_0]$  [Hz] に制限されているならば, 標本化周波数  $f_s = 2f_0$  以上で標本化すれば, その標本値系列  $f(mT_0)$  ( $m = 0, \pm 1, \pm 2, \dots$ ) より完全に  $f(t)$  が定まる. ただし, 標本化間隔  $T_0 = 1/2f_0$  である. □

この定理により周波数帯域制限のある連続関数  $f(t)$  は  $T_0$  [sec] 毎の標本値  $f(mT_0)$  で完全に復元されることがわかったが, 振幅値  $f(mT_0)$  は依然として連続値である. そこで, 音声・画像などの振幅が連続な信号を人間の耳や目が判別不可能な (あるいは歪を許容して) 振幅のレベル値に分解し, 離散値として表わす. これを**量子化**という. 量子化幅を  $\Delta$  とすると, 整数  $l$  に対し

$$\left(l - \frac{1}{2}\right) \Delta \leq f(mT_0) < \left(l + \frac{1}{2}\right) \Delta \implies f(mT_0) \rightarrow l\Delta \quad (7.12)$$



とする操作が量子化である。以上の結果、標本化と量子化により連続信号はある条件<sup>2</sup>の下に時間軸・振幅軸ともに離散化され、アナログ信号はデジタル信号に変換される。このときの信号空間を図 7.2 に示す。

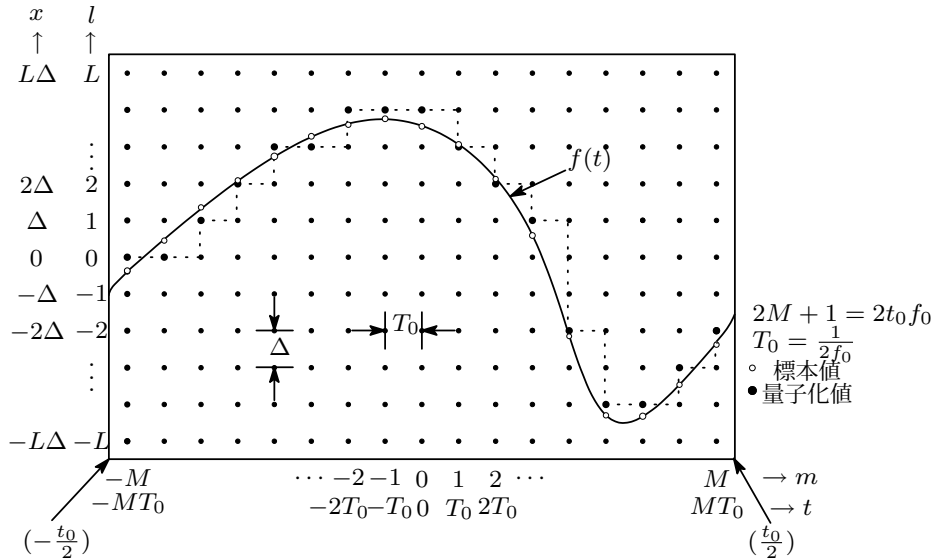


図 7.2: 標本化と量子化による信号空間

## (2) 情報源符号化とハフマン符号

情報源を式 (7.5) で与えた完全事象系  $\mathcal{A}$  とする。  $\mathcal{A}$  からの出力<sup>3</sup>  $a_i (i = 1, 2, \dots, n)$  を 2 元記号列  $\mathbf{v}_i \in \{0, 1\}^{l_i}$  に写像する<sup>4</sup>。これを情報源符号化  $C_s$  という。すなわち

$$C_s : a_i \in \mathcal{A} \rightarrow \mathbf{v}_i \in \{0, 1\}^{l_i}$$

である。  $l_i$  を記号  $a_i$  に対する符号長といい、平均符号長  $\bar{l}$  を次式で与える。

$$\bar{l} = \sum_{i=1}^n p_i l_i \quad (7.13)$$

このとき、次の定理が成り立つ。

<sup>2</sup>  $t_0 f_0 \gg 1, |\Delta| \ll 1$ 。ただし、  $t_0$  は  $f(t) = 0 (|t_0| \geq t)$  となる値である。

<sup>3</sup> 事象  $a_i$  を出力記号と考える。

<sup>4</sup> 一般には多元記号である。

[定理 7.3] (情報源符号化定理) 情報源  $\mathcal{A}$  のエントロピーを  $H(\mathcal{A})$  [bits/symbol] とする. 平均符号長  $\bar{l}$  が次式を満たす符号化が可能である.

$$H(\mathcal{A}) \leq \bar{l} < H(\mathcal{A}) + 1 \quad (7.14)$$

□

このように, エントロピー  $H(\mathcal{A})$  は平均符号長  $\bar{l}$  の下界を与える. 次に示す **ハフマン (Huffman) 符号** は記号数  $n$  が有限の情報源において最小の平均符号長  $\bar{l}$  を与える.

[ハフマン符号]

- ①  $n$  個の記号を確率の大きい順に並べる.
- ② 確率の最も小さい記号から 2 個をとってそれぞれに “0” と “1” を与え, その確率和を計算する.
- ③ その 2 個を 1 記号とみなして確率の大きい順に並べ直す. ただし, おきかえた記号の確率は 2 個の記号の確率和とする.
- ④ ②③の操作を確率 1 の単一記号になるまで続ける.
- ⑤ 各段階で割り当てた “0”, または “1” を単一記号から逆にたどって  $n$  個の記号に対応する符号語をつくる.

[例 7.2] (ハフマン符号の例)

$n=6$  の場合のハフマン符号の例を図 7.3 に示す. □

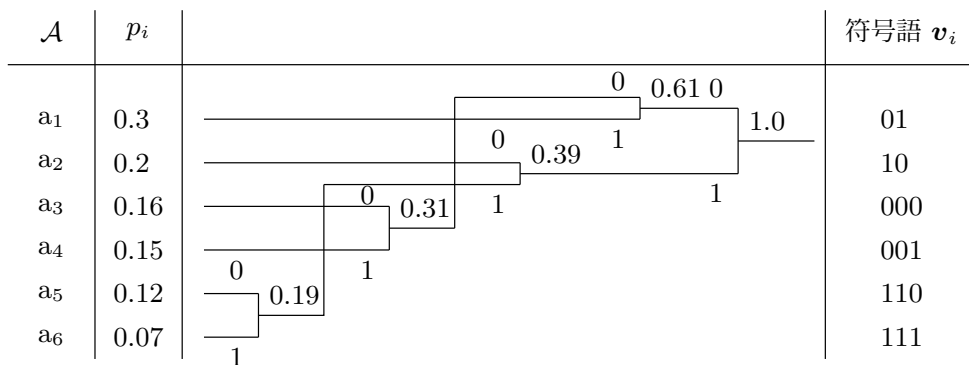


図 7.3: 情報源  $\mathcal{A}$  の 2 元ハフマン符号

なお, コンピュータにおけるファイルの圧縮に用いられる UNIX のコマンドの `compress` や MS-DOS の `ARC`, `PKZIP`, `LHA` などは LZ 符号とよばれる (情報源の統計的性質によらない) **ユニバーサル符号** である.

### 7.3 通信路符号化

#### (1) 通信路容量

通信路の入力事象系を式 (7.5) の  $\mathcal{A}$ , 出力事象系を式 (7.8) の  $\mathcal{B}$  とする. 通信路の特性が条件付確率  $\Pr(b_j|a_i) (i = 1, 2, \dots, n; j = 1, 2, \dots, m)$  で与えられているとき, 通信路容量  $C$  は式 (7.9) の相互情報量  $I(\mathcal{A}; \mathcal{B})$  の確率ベクトル  $\mathbf{p} = (p_1, p_2, \dots, p_n)$  に対する最大値で与えられる. すなわち

$$C = \max_{\mathbf{p}} I(\mathcal{A}, \mathcal{B}) \quad [\text{bits/symbol}] \quad (7.15)$$

である.

[例 7.3] (2元対称通信路の通信路容量)

誤り確率  $\varepsilon$  の 2元対称通信路 (図 8.1 参照) の通信路容量  $C$  は式 (7.15) において  $\mathbf{p} = (\frac{1}{2}, \frac{1}{2})$  のとき得られ, 式 (7.7) を用いて次のように求められる.

$$C = 1 - H_b(\varepsilon) \quad [\text{bits/symbol}] \quad (7.16)$$

□

#### (2) 通信路符号化

長さ  $K$  の 2元記号ベクトル (情報記号系列)  $\mathbf{w} = (w_1, w_2, \dots, w_K) (w_i \in \{0, 1\})$  を長さ  $N$  の符号語  $\mathbf{x} = (x_1, x_2, \dots, x_N) (x_i \in \{0, 1\})$  に写像する<sup>5</sup>. これを通信路符号化  $C_c$  という. すなわち

$$C_c : \mathbf{w} \in \{0, 1\}^K \rightarrow \mathbf{x} \in \{0, 1\}^N \quad (N \geq K) \quad (7.17)$$

である. このとき, 情報伝送速度  $R$  は

$$R = K/N \quad [\text{bits/symbol}] \quad (7.18)$$

で与えられる.

[定理 7.4] (通信路符号化定理) 通信路の通信路容量を  $C$  [bits/symbol] とする. 情報伝送速度  $R$  [bits/symbol] が  $R < C$  ならば, 復号誤り確率  $\Pr(\mathcal{E}) \rightarrow 0 (N \rightarrow \infty)$  となる符号長  $N$ , 情報伝送速度  $R$  の符号が存在する. □

<sup>5</sup>もちろん, 多元記号 ( $q$ 元,  $q > 2$ ) で符号化することも可能である.

## 演習問題

[7.1] 例 7.2 において, エントロピー  $H(\mathcal{A})$  と平均符号長  $\bar{l}$  を求め両者を比較せよ.

[7.2] 次の情報源  $\mathcal{A}$  のハフマン符号を求めよ. エントロピー  $H(\mathcal{A})$ , 平均符号長  $\bar{l}$  はいくらか.

$$\mathcal{A} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 0.5 & 0.25 & 0.125 & 0.125 \end{bmatrix}$$

[7.3] 連続関数  $f(t)$  の周波数スペクトル  $F(\omega)$  が  $F(\omega) = 0$  ( $|\omega| > \omega_0, \omega_0 = 2\pi f_0$ ) であるとき, 式 (7.11) を導け.

[7.4] 完全事象系  $\mathcal{A}, \mathcal{B}$  を  $\mathcal{A} = \begin{bmatrix} a_1, a_2, \dots, a_n \\ p_1, p_2, \dots, p_n \end{bmatrix}$   $\mathcal{B} = \begin{bmatrix} b_1, b_2, \dots, b_n \\ q_1, q_2, \dots, q_n \end{bmatrix}$  とする.

## (1) Kulback-Leibler 情報量

$$I(\mathbf{p} \parallel \mathbf{q}) = \sum_{i=1}^n p_i \log p_i - \sum_{i=1}^n p_i \log q_i \geq 0 \quad (\text{q.7.4.1})$$

が成り立つこと, 等号は  $p_i = q_i (i = 1, 2, \dots, n)$  のとき, かつそのときに限り成り立つことを示せ.

$$(2) H(\mathcal{A}) = -\sum_{i=1}^n p_i \log p_i \leq \log n \quad (\text{q.7.4.2})$$

および等号は  $p_i = 1/n$  のとき, かつそのときに限り成り立つことを示せ.

## 参考文献

1. 三根久, 情報理論入門, 朝倉書店, 昭 39 年.
2. 関英男, 情報理論, オーム社, 昭 44 年.
3. 赤攝也, 伊藤正夫, 後藤英一, 広瀬健編, 情報処理のための数学, 共立出版, 昭 50 年.
4. 有本卓, 情報理論, 共立出版, 昭 51 年.
5. 今井秀樹, 情報理論, 昭晃堂, 昭 59 年.
6. 笠原正雄, 田崎三郎, 小倉久直, 情報理論—基礎と応用, 昭晃堂, 昭 60 年.
7. 汐崎陽, 情報・符号理論の基礎, オーム社, 平成 3 年.
8. 大石進一, 例にもとづく情報理論入門, 講談社, 平成 5 年.
9. 橋本猛, 情報理論, 培風館, 平成 9 年.
10. 平澤茂一, 情報理論入門, 情報数理論シリーズ A-6, 培風館, 2000 年.

## 8 符号理論

1950年、R.W.Hamming は後にハミング符号とよばれる単一(記号)誤り訂正符号を発見した。この符号は当時、信頼性の低い素子で作られたコンピュータの主メモリ装置のために考え出された今日のフォールトトレラントコンピューティングのためであった。同時期に M.J.E.Golay によるゴレイ符号も発見され一気に活発な研究が開始された。

符号理論(誤り訂正符号の理論)は Shannon の通信路符号化が対象とする情報の高信頼化、すなわち誤りを訂正あるいは検出する多くの具体的符号化方法を与える。組合せ数学など抽象代数学を用いて、符号の構成法、および代数的復号法を与えることが主眼である。

誤り訂正符号の応用例はコンピュータ間通信(パケット交換)・深宇宙通信・CD(Compact Disc)・コンピュータの主メモリなど身の周りに数多い。考え方を理解するために、次の例を示しておく。なお、ここでも式(7.17)の符号化  $C_c$  を仮定する。

[例 8.1] (学籍番号(modulus 11))

学籍番号などの転記ミスや手書き文字の認識誤りを発見するために、modulus11 というチェックディジット方式がある。例えば、学部・入学年度・学科・個人番号を記号化して

$$G7H017 \rightarrow G7H017-0 \quad (8.1)$$

とする。式(8.1)の右側最後尾の-0はチェックディジットである<sup>1</sup>。これを式(7.17)において

$$\begin{aligned} \mathbf{w} &= (w_1, w_2, \dots, w_6) \rightarrow \mathbf{x} = (x_1, x_2, \dots, x_6, x_7) \\ w_i &= x_i \quad (i = 1, 2, \dots, 6) \end{aligned} \quad (8.2)$$

と表す。このとき、英文字を  $A \rightarrow 1$ ,  $B \rightarrow 2$  のように数字に置き換えると

$$2x_1 + 3x_2 + 4x_3 + 5x_4 + 6x_5 + 7x_6 + x_7 = 0 \pmod{11} \quad (8.3)$$

となっている。ただし、 $x_7 = 0$  のとき  $x_7 = 11 \rightarrow 1$  に、 $x_7 = 10$  のとき  $x_7 = 0$  にする(式(8.1)の例は  $x_7 = 10 \rightarrow x_7 = 0$  とした場合である)。これにより、かなり高い確率で文字の入れ換え誤りや、6と4の読み違いなどが検出できる。ただし、符号語  $\mathbf{x}$  が満足する条件式は1つなので誤り訂正能力は望めない。□

例では人為的ミス発見のため10進の符号を扱ったが、以降2元記号により構成される2元符号を仮定し、通信路は図8.1に示す誤り確率  $\varepsilon$  の2元対称通信路とする。なお、 $q$ 元符号への拡張はさほどむずかしくない。ただし、 $q$ は素数のべき乗とする。

<sup>1</sup>小売店の商品に付けられているバーコードも10進13桁、うち1桁がチェックディジットで JAN コードとよばれている。

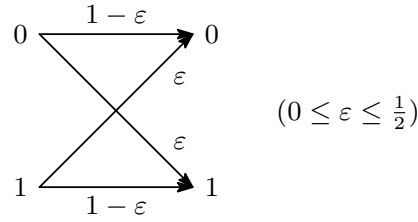


図 8.1: 誤り確率  $\varepsilon$  の 2 元対称通信路のモデル

## 8.1 誤り訂正・検出のしくみ

### (1) ハミング距離

2 元記号を要素とする長さ  $n$  の二つのベクトル  $\mathbf{x}_i$  と  $\mathbf{x}_j$  を次のように与える.

$$\begin{aligned}
 \mathbf{x}_i &= (x_{i1}, x_{i2}, \dots, x_{in}) \\
 \mathbf{x}_j &= (x_{j1}, x_{j2}, \dots, x_{jn})
 \end{aligned}
 \tag{8.4}$$

ここで

$$x_{im}, x_{jm} \in GF(2) \quad (m = 1, 2, \dots, n) \tag{8.5}$$

である.  $\mathbf{x}_i, \mathbf{x}_j$  のハミング距離  $D_H(\mathbf{x}_i, \mathbf{x}_j)$  は次式で定義される.

$$D_H(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^n d_H(x_{im}, x_{jm}) \tag{8.6}$$

ここで

$$d_H(a, b) = \begin{cases} 0, & a = b; \\ 1, & a \neq b \end{cases} \tag{8.7}$$

である. すなわち, ハミング距離は 2 つのベクトルの互いに異なる要素の数である.

2 元  $(n, k, d)$  符号  $C$  は 2 元ベクトル  $\mathbf{x}_i$  ( $i = 1, 2, \dots, M$ ) の集合である. ここで,  $M = 2^k$  であり,  $\mathbf{x}_i$  は符号語とよばれる. また,  $n$  は符号長,  $k$  は情報記号数,  $d$  は次式で定義される最小距離である.

$$d = \min_{\substack{1 \leq i, j \leq M \\ i \neq j}} D_H(\mathbf{x}_i, \mathbf{x}_j) \tag{8.8}$$

また, 符号化比率  $r$  は次式で与えられる.

$$r = \log_2 M/n = k/n \tag{8.9}$$

### (2) ハミング重み

$\mathbf{x}_i$  のハミング重み  $W_H(\mathbf{x}_i)$  は  $n$  記号中、非ゼロ記号  $x_{im}$  の数である。すなわち

$$W_H(\mathbf{x}_i) = \sum_{m=1}^n w_H(x_{im}) \tag{8.10}$$

ここで

$$w_H(a) = \begin{cases} 0, & a = 0; \\ 1, & a = 1 \quad (a \neq 0) \end{cases} \tag{8.11}$$

である。この結果、容易に次式を得る。

$$D_H(\mathbf{x}_i, \mathbf{x}_j) = W_H(\mathbf{x}_i - \mathbf{x}_j). \tag{8.12}$$

(3) 誤り訂正・検出能力

[定理 8.1]  $(n, k, d)$  符号は

- (1)  $d - 1$  個以下のすべての誤りを検出できる。
- (2)  $t$  個以下のすべての誤りを訂正できる。ここで、 $2t + 1 \leq d$  である。

□

定理 8.1 より、容易に  $(n, k, d)$  符号が  $t'$  個以下のすべての誤りパターンを訂正でき、 $d'$  個以下のすべての誤りパターンを検出できる条件は、 $d \geq t' + d' + 1$  ( $d' > t'$ ) であることがわかる。このとき、距離  $d$  の 2 つの符号語  $\mathbf{x}_i, \mathbf{x}_j$  は図 8.2 のようになる。

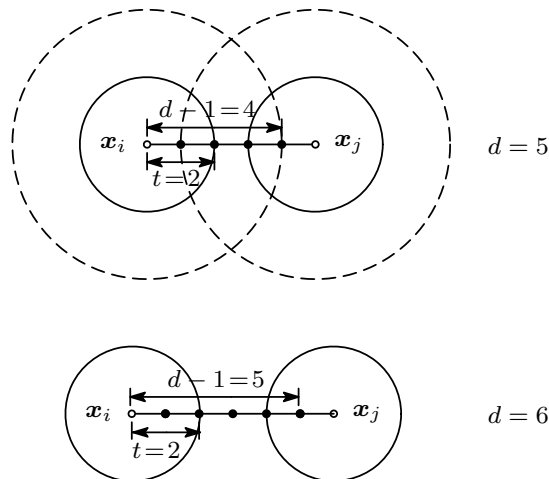


図 8.2: 符号語の位置と誤り訂正・検出可能な個数

## 8.2 重要な符号

### (1) 線形符号

2元  $(n, k, d)$  線形符号は  $F^n = \{0, 1\}^n$  の線形部分空間である。ただし、演算規則は表 8.1 にしたがう。これをガロア体  $GF(2)$  という。  $\mathbf{x}_i$  と  $\mathbf{x}_j$  が共に線形符号の符号語とすると、  $\mathbf{x}_i + \mathbf{x}_j = \mathbf{x}_l$  もその符号語でなければならない。したがって、2元線形符号では  $W_H(\mathbf{x}_l) = D_H(\mathbf{x}_i, \mathbf{x}_j)$  が成り立つ<sup>2</sup>。線形符号は必ず全ゼロベクトルを符号語として持つから、次の定理が得られる。

表 8.1: 加法と乗法 ( $GF(2)$ ) の演算

$a$	$b$	$a + b$	$a \cdot b$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

[定理 8.2] 線形符号の最小距離は非ゼロ符号語の最小重みに等しい。

□

線形部分空間の次元が  $k$  のとき、  $(n, k, d)$  線形符号から互いに線形独立な  $k$  個の符号語をとり出し、これを  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$  とする。このとき、生成行列  $G$  は

$$G = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_k \end{bmatrix} \quad (8.13)$$

で与えられる。ここで、  $G$  は  $k \times n$  の行列で、その階数 (ランク) は  $k$  である。通常の行列の行操作により、  $(n, k, d)$  線形符号の生成行列  $G$  は次の正準形で与えられる。

$$G = [I_k, P] \quad (8.14)$$

<sup>2</sup> $GF(2)$  では  $-1 = +1, -\mathbf{x} = +\mathbf{x}$  であることに注意。



ここで、 $I_k$  は  $k \times k$  の単位行列、 $P$  は  $k \times (n - k)$  の行列である。

符号化される長さ  $k$  の情報記号系列 (データベクトル) を  $\mathbf{w}$  とすると、これに 1 対 1 に対応する符号語  $\mathbf{x}$  は次式で与えられる。

$$\begin{aligned}\mathbf{w} &= (w_1, w_2, \dots, w_k) \\ \mathbf{x} &= \mathbf{w}G = (x_1, x_2, \dots, x_n)\end{aligned}\tag{8.15}$$

ここで、 $G$  は  $k \times k$  の単位行列を持つから、 $w_m = x_m$  ( $m = 1, 2, \dots, k$ ) である。このように、 $k$  個の情報記号と  $n - k$  個の検査記号が区別できるような符号を組織符号とよぶ。

生成行列  $G$  が与えられたとき、 $GH^T = 0$  が成り立つ  $H$  をパリティ検査行列という。

[定理 8.3] 生成行列  $G$  が式 (8.14) で与えられる  $(n, k, d)$  線形符号のパリティ検査行列  $H$  は次式で与えられる。□

$$H = [-P^T, I_{n-k}]\tag{8.16}$$

ここで、 $I_{n-k}$  は  $(n - k) \times (n - k)$  の単位行列、 $P^T$  は  $(n - k) \times k$  の式 (8.14) で与えた行列  $P$  の転置行列である。

[定義 8.1] 受信語  $\mathbf{y}$  のシンドローム  $\mathbf{s}$  は次式で与えられる。

$$\mathbf{s} = \mathbf{y}H^T\tag{8.17}$$

□

いま、 $\mathbf{x}$  が送信されたとし、これに通信路で雑音ベクトル  $\mathbf{e}$  が加えられたとする。すなわち

$$\mathbf{y} = \mathbf{x} + \mathbf{e}\tag{8.18}$$

ここで

$$\mathbf{e} = (e_1, e_2, \dots, e_n), \quad e_m \in GF(2) \quad (m = 1, 2, \dots, n)\tag{8.19}$$

である。  $GH^T = 0$  であるから、次式が得られる<sup>3</sup>。

$$\mathbf{s} = \mathbf{y}H^T = (\mathbf{x} + \mathbf{e})H^T = \mathbf{e}H^T\tag{8.20}$$

<sup>3</sup>式 (8.20) において、常に  $\forall \mathbf{w}, \mathbf{w}GH^T = \mathbf{x}H^T = 0$  である。この式は例 8.1, 学籍番号において、 $\mathbf{x}H^T = 0 \pmod{11}$  ただし、 $H = [2, 3, 4, 5, 6, 7, 1]$  に対応していることに注意する。

式 (8.20) により, シンドローム  $s$  は  $e_m$  が 1 に対応する  $H$  の列の和の転置であることがわかる.

**[定理 8.4]** 2 元  $(n, k, d)$  線形符号を考える. そのパリティ検査行列  $H$  は任意の  $d - 1$  個およびそれ以下の列が  $GF(2)$  で線形独立<sup>4</sup>である  $(n - k) \times n$  の行列である. □

## (2) ハミング符号

2 元ハミング符号は重要で最もよく知られた符号であり, 数少ない**完全符号**の一つである.

$m$  行  $(2^m - 1)$  列のパリティ検査行列  $H$  を考える. ただし,  $m \geq 2$  である.  $(n, k, d)$  ハミング符号はすべて 0 のパターンを除く長さ  $m$  のすべての可能なパターンを列ベクトルとするパリティ検査行列<sup>5</sup>  $H$  で定義される符号である. 定理 8.4 より,  $(n, k, d)$  **ハミング符号**はすべての単一誤りを訂 ( $d = 3$ ) 正できる. ここで, パラメータは次式で与えられる.

$$n = 2^m - 1, \quad k = 2^m - m - 1, \quad d = 3 \quad (8.21)$$

これに偶数パリティ検査記号を付加して得られる  $(n, k, d)$  **拡張ハミング符号**は単一誤り訂正・二重誤り検出 ( $d = 4$ ) ができる. ここで, パラメータは次式で与えられる.

$$n = 2^m, \quad k = 2^m - m - 1, \quad d = 4 \quad (8.22)$$

## (3) 巡回符号

線形かつ符号語の巡回置換もまた符号語であるような符号  $C$  を**巡回符号** (cyclic code) とよぶ. すなわち

$$\mathbf{x} = (x_0, x_1, x_2, \dots, x_{n-1}) \in C \quad (8.23)$$

のとき

$$\mathbf{x}^C = (x_{n-1}, x_0, x_1, \dots, x_{n-2}) \in C \quad (8.24)$$

<sup>4</sup>列ベクトルの和が非ゼロ. なお,  $d$  個以上の列ベクトルが線形従属のとき, 最小距離は  $d$  以下である.

<sup>5</sup>任意の  $d - 1 = 2$  個の列ベクトルは異なる.

である。ここで、 $x_i \in GF(2)$  ( $i = 0, 1, \dots, n-1$ ) である。代数的記述を容易にするために、ベクトル  $\mathbf{x}$  を多項式  $x(z)$

$$x(z) = x_0 + x_1z + \dots + x_{n-1}z^{n-1} \quad (8.25)$$

に対応させる。ここで、 $x(z)$  は  $R_n = F[z]/(z^n+1)$  の元、すなわち  $(\text{mod } z^n+1)$  の多項式環の要素である。いま、 $x(z)$  に  $z$  を乗ずる。その結果

$$\begin{aligned} zx(z) &= x_0z + x_1z^2 + \dots + x_{n-1}z^n \\ &= x_{n-1} + x_0z + \dots + x_{n-2}z^{n-1} \pmod{z^n+1} \end{aligned} \quad (8.26)$$

となり、これは  $\mathbf{x}$  を右に巡回置換したベクトル  $\mathbf{x}^C$  に対応する。代数的にいえば、環  $R_n$  のイデアル  $I$  はもし  $x(z) \in I$  ならば  $zx(z) \in I$  であるような  $R_n$  の線形部分空間である。したがって、 $x(z) \in I$  ならば、 $a(z) \in R_n$  に対し、 $a(z) \times x(z) \in I$  である。よって、長さ  $n$  の巡回符号は環  $R_n$  のイデアルであるという。符号長  $n$ 、情報記号数  $k$  の巡回符号  $C$  において、情報多項式  $w(z)$ 、生成多項式  $G(z)$ 、 $\deg G(z) = n - k$  とするとき、符号多項式  $x(z) \in C$  は次式のように唯一に表現される。

$$x(z) = w(z)G(z) \quad (8.27)$$

ここで、 $\deg w(z) < k$  である。なお、組織的巡回符号の符号語  $x(z)$  は

$$x(z) = w(z)z^{n-k} + r(z) = q(z)G(z) \quad (8.28)$$

として生成される。

## 演習問題

[8.1] 例 8.1 で示した学籍番号において

- (1) G9H045-2 は正しいか
- (2) G9H0 □ 3-6 において □ を求めよ
- (3) 式 (8.1) で個人番号の 017 を 071 と入れかえ誤りが生じたとき、これを検出できるか

ただし、 $A \rightarrow 1, B \rightarrow 2, \dots, G \rightarrow 7, H \rightarrow 8, \dots$  のように読みかえる。

[8.2] 生成行列  $G$ 、パリティ検査行列がそれぞれ式 (8.14), (8.16) で与えられるとき、 $GH^T = 0$  であることを示せ。

[8.3] 2元 (7,4,3) ハミング符号について

- (1) その生成行列  $G$ 、およびパリティ検査行列  $H$  の例を示せ。
- (2) 情報記号系列  $\mathbf{w} = (0101)$  のときの符号語  $\mathbf{x}$  を求めよ。

- (3)  $x$  を送信し, 誤り  $e_1 = (0001000)$ , および  $e_2 = (0010001)$  のとき, 受信系列  $y_1, y_2$  およびそのシンδροーム  $s_1, s_2$  を求めよ.
- (4)  $s_1, s_2$  を用いて誤りを訂正せよ.

### 参考文献

1. 宮川洋, 岩垂好裕, 今井秀樹, 符号理論, 昭晃堂, 1973 年.
2. 嵩忠雄, 都倉信樹, 岩垂好裕, 稲垣康善, 符号理論, コロナ社, 1975 年.
3. 平澤茂一, 西島利尚, 符号理論入門, 情報数理論シリーズ A-3, 培風館, 1999 年.
4. 平澤茂一, 情報理論入門, 情報数理論シリーズ A-6, 培風館, 2000 年.

# 9

## 暗号理論

映画 2001 年宇宙の旅では宇宙船に HAL 9000 というコンピュータが登場する，“HAL”は“IBM”の各文字を ABC 順に 1 文字左シフトすると現れる。これはシーザー暗号とよばれる最も単純な古典暗号の 1 つである<sup>1</sup>。暗号の歴史は古く、ギリシャ・ローマ時代にさかのぼる。古典暗号の多くは仲間どうしの通信の秘匿が目的で、主として外交・軍事用であり余り明るいイメージはない。第 2 次世界大戦で暗号が陰の兵器として果たした役割は大きく、ドイツのエニグマ暗号、日本の九七式暗号（パープル暗号）、アメリカ陸軍の M209 などが活躍している。

しかし、暗号は宝島の地図や謎解きなど推理・探偵・冒険小説にも出てくる。そればかりか、ノストラダムスの予言や聖書にもかくされた情報がうめ込まれているという。さらに、情報のデジタル化にともなう複製（コピー）防止（著作権保護）やインターネット時代の電子商取引・電子現金などの実用化に向けて、暗号技術は情報秘匿・個人認証の情報セキュリティ社会を担うものとして注目されている。なお、近代暗号は仲間どうしの通信としてだけでなく、主として商用を目的とした情報化社会の要の技術の役割をもつために、広く一般ユーザが参加できるようアルゴリズムを公開した点が従来の古典暗号と大きく異っている。

近代暗号には 1977 年米国標準暗号 **DES**(Data Encryption Standard) で代表される秘密鍵暗号系と 1976 年、W.Diffie と M.Hellman により提案された公開鍵暗号系がある。特に後者は鍵の一部まで公開するという画期的なもので、R.L.Rivest, A.Shamir と L.Adelman によって考え出された **RSA 暗号**がよく知られている。

### 9.1 暗号系のモデルと暗号強度

#### (1) 暗号系のモデル

暗号系のモデルを図 9.1 に示す。情報源  $M$  から出力された通常の誰にも意味が理解できる平文を  $m$ 、これを暗号化鍵  $K_E$  を用いて暗号器により暗号化し意味が理解できない暗号文  $c$  に変換し通信路に送り出す。通信路には誤りはないものとし、復号器の入力  $c$  から復号鍵  $K_D$  により復号し元の平文  $m$  を受信

---

<sup>1</sup>ここでは、手作業あるいは機械式によって暗号文を生成するものを古典暗号とよび、電子化、デジタル化、コンピュータ化などと整合したものを近代暗号とよぶことにする。

者  $\hat{\mathcal{M}}$  に出力する. これを次のように表わす.

$$\begin{aligned} \text{暗号化 } C : c &= C_{K_E}(m) \\ \text{復号 } D : m &= D_{K_D}(c) \end{aligned} \quad (9.1)$$

ここで

$$m = D_{K_D}(C_{K_E}(m)) \quad (9.2)$$

が成立しなければならない. ここでは,  $m, c$  とも 2 元記号または 10 元記号とし, 数十ビット以上, あるいは 10 進数数百桁のブロックごとに暗号化, 復号するブロック暗号を示すが, 1 ビットあるいは 10 進数 1 桁ごとに逐次暗号化・復号するストリーム暗号もある.

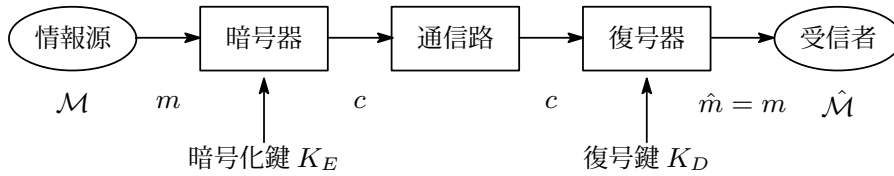


図 9.1: 暗号系のモデル

## (2) 情報量と暗号強度

まず, 式 (9.1), (9.2) において  $K_E = K_D = K$  とする. すなわち, 暗号化鍵と復号鍵は同一で共に  $K$  とする **共通鍵暗号 (秘密鍵暗号)** を仮定する. ここで, 鍵の集合を  $\mathcal{K}$  で表わす.  $K \in \mathcal{K} = \{K_1, K_2, \dots, K_{\|\mathcal{K}\|}\}$  である. このとき鍵のエントロピー  $H(\mathcal{K})$  は

$$H(\mathcal{K}) = - \sum_{i=1}^{\|\mathcal{K}\|} \Pr(K_i) \log \Pr(K_i) \quad (9.3)$$

である.

**[定理 9.1]** 長さ  $L$  の暗号文  $c \in \mathcal{C}^L$  を知ったときの鍵  $K_i \in \mathcal{K}$  のもつ平均情報量  $H(K|\mathcal{C}^L)$  は適当な条件<sup>2</sup>の下に次式で与えられる.

$$H(K|\mathcal{C}^L) = H(\mathcal{K}) - LR(\mathcal{M}) \quad (9.4)$$

ここで,  $R(\mathcal{M})$  は平文  $m \in \mathcal{M}^L$  の 1 記号当りの平均冗長度である.  $\square$

<sup>2</sup>鍵  $K_i$  を知っていれば  $H(\mathcal{C}^L|\mathcal{K}) = H(\mathcal{M}^L)$ , 一方  $H(\mathcal{K}|\mathcal{C}^L) + H(\mathcal{C}^L) = H(\mathcal{C}^L|\mathcal{K}) + H(\mathcal{K})$  である.  $R(\mathcal{M}^L) = L \log \|\mathcal{M}\| - H(\mathcal{M}^L)$ ,  $H(\mathcal{C}^L) = L \log \|\mathcal{C}\|$ ,  $\mathcal{M} = \mathcal{C}$  と仮定し,  $R(\mathcal{M}^L) = L \log \|\mathcal{M}\| - LH(\mathcal{M}) = LR(\mathcal{M})$  とすればよい.

式 (9.4) より暗号強度 (暗号の安全性) に関し次のことがわかる.

- (1)  $H(K)$  が大きい (もし, 鍵  $K_i$  が等確率で用いられるとすれば, 鍵の集合の要素数  $\|K\|$  が大きい) ほど安全である.
- (2)  $H(K|C^L) = 0$  となる長さ  $L_1$  を一義長 (判別距離) という.  $L_1 = H(K)/R(M)$  であるから, 平文の冗長度が小さいほど安全である.
- (3)  $H(K) > LR(M)$  のとき情報理論的に安全, すなわち暗号文の長さ  $L$  は小さいほど安全である.

これに対し, 9.2 節以下で示す暗号系は解読のために要するコンピュータ資源の点から安全性を保証するもので, 計算論的に安全という.

## 9.2 秘密鍵暗号系

**秘密鍵暗号** (慣用暗号, 対称鍵暗号, 共通鍵暗号ともいう) は送信者 (情報源) と受信者があらかじめ共通の秘密情報 (秘密鍵  $K$ ) を共有し, この情報をもとに暗号化や認証を行う方式である. したがって, 図 9.1 および式 (9.1), (9.2) において  $K_E = K_D = K$  である. 表 9.1 に古典暗号における暗号アルゴリズムの分類を示す. アルゴリズムを公開する近代暗号では基本的には鍵に基づいて転字・換字を繰返し適用する<sup>3</sup>. 記号はビット, バイト, 英数字などを単位としている. アルゴリズムが簡単で高速処理が可能である. 秘密鍵暗号で有名なものは, 1977 年米国 NBS により標準暗号として制定された **DES** である.

表 9.1: 古典暗号アルゴリズムの分類と例

方式	アルゴリズム	鍵	例
換字法 (置換法)	平文の各記号を他の記号に置き換える方法	シフト文字数 (ブロック長)	シーザー暗号 多書式 (ブロック化) シーザー暗号
転字法 (転置法)	平文の各記号の順序を入れかえる方法	交錯寸法 転字規則図	スキュタレー暗号 蜘蛛 (くも) の経路

第 2 次世界大戦中に外交・軍用に用いられたエニグマ暗号やパープル暗号は原理的には換字法に分類される. いずれも 10 億以上の周期をもつロータの初期位置を鍵とし, 次々と換字表を求める数字を生成させる方法である.

<sup>3</sup>1949 年, C.E.Shannon は転字・換字を繰返し用いることにより, 暗号強度が上がることを明らかにしている.

### 9.3 公開鍵暗号系

**公開鍵暗号** (非対称鍵暗号ともいう) は暗号化鍵  $K_E$  と復号鍵  $K_D$  を一対ずつ作り, 暗号化鍵を公開, 復号鍵を秘密にする暗号方式である. 送信者 A は公開リスト (電話帳のようなもの) 上の受信者 B の暗号化鍵  $K_E$  を用いて暗号文を作成し, 受信者 B は自分だけが知っている復号鍵  $K_D$  (これは秘密とする) を用いて暗号文を復号し平文を得ることができる. 公開鍵暗号系では暗号化鍵  $K_E$  と復号鍵  $K_D$  が異なり, 公開された  $K_E$  から秘密にする  $K_D$  が容易に求まらない. この仕組みを作るには**一方向性関数**が重要な役割をはたす<sup>4</sup>.

公開鍵暗号では, 暗号解読の難しさは数論の難問を解く難しさに対応している. 表 9.2 に代表的な数論の難問とこれを用いた暗号系を示す<sup>5</sup>.

表 9.2: 数論の難問題 (一方向性関数) と応用例  
( $\rightarrow$  は容易,  $\dashrightarrow$  は困難を示す)

数論の難問題	応用例	備考	条件
素因数分解問題	RSA 暗号, RSA 署名 Williams 暗号 Goldwasser-Micali-Yao 署名	$n = p \cdot q$ $(p, q) \rightarrow n$ $n \dashrightarrow (p, q)$	$p, q$ : 素数
離散対数問題	Elgamal 暗号 Elgamal 署名	$y = \alpha^x \pmod{p}$ $(\alpha, x, p) \rightarrow y$ $(\alpha, y, p) \dashrightarrow x$	$p$ : 素数, $0 < \alpha \leq p - 1$ , ただし $\alpha \in GF(p)$ の原始根
平方剰余問題	Rabin 暗号 Fiat-Shamir 法 (ゼロ知識証明法)	$y = x^2 \pmod{n}$ $(x, n) \rightarrow y$ $(y, n) \dashrightarrow x$	$p, q$ : 素数, $n = pq$
ナップザック問題	Merkle-Hellman 暗号 Graham-Shamir 暗号 Chor-Rivest 暗号	$s = \mathbf{a}\mathbf{x}^T$ $\mathbf{a} = (a_1, a_2, \dots, a_n)$ $\mathbf{x} = (x_1, x_2, \dots, x_n)$ $(\mathbf{a}, \mathbf{x}) \rightarrow s$ $(s, \mathbf{a}) \dashrightarrow \mathbf{x}$	$s, a_i$ : 正整数, $x_i \in \{0, 1\}$
線形符号の 復号問題	McEliece 暗号	$G' = SGP$ $(S, G, P) \rightarrow G'$ $G' \dashrightarrow (S, G, P)$	$G$ : 生成行列 ( $k \times n$ ) $S$ : 正則行列 ( $k \times k$ ) $P$ : 置換行列 ( $n \times n$ )

#### (1) RSA 暗号

RSA 暗号は, その安全性を大きな整数 (例えば, 10 進数 200 桁) の素因数分解問題を解く困難さに根拠をおいている.

<sup>4</sup>関数  $f(\cdot)$  が一方向性であるということは, 順方向  $f(\cdot)$  の計算は容易 (多項式時間で計算可能) であるが, 逆方向  $f^{-1}(\cdot)$  の計算は (多項式時間では) 困難であることをいう.

<sup>5</sup>ただし, 既に解読された暗号もある.



**RSA 暗号のアルゴリズム****準備**

- (1) 2つの大きな素数  $p, q$  を選び, RSA 合成数  $n = pq$  を求める.  $p, q$  は秘密にしておく.
- (2) オイラー関数  $\phi(n) = (p-1)(q-1)$  を計算し<sup>6</sup>

$$\text{GCD}(d, \phi(n)) = 1 \quad (9.5)$$

$$ed = 1 \pmod{\phi(n)} \quad (9.6)$$

を満たす対  $(e, d)$  を求める. すなわち,  $\phi(n)$  と互いに素な整数  $d$  を求め<sup>7</sup>,  $\phi(n)$  を法とする演算の下に  $d$  の逆数を  $e$  とする<sup>8</sup>.

- (3) ここで

$$K_E \text{ (暗号化鍵)} : (e, n) \quad (9.7)$$

$$K_D \text{ (復号鍵)} : (d, n)$$

とし,  $K_E$  を公開し,  $K_D$  を秘密に保つ<sup>9</sup>.

**暗号化と復号**

- (1) 送信者 A はメッセージ (平文)  $m$  を  $0 \leq m \leq n-1$  の整数で表現し, 受信者 B の公開鍵  $K_E$  を用いて暗号文  $c$  を

$$\text{暗号化} : c = m^e \pmod{n} \quad (9.8)$$

とする<sup>10</sup>. もちろん, 暗号文  $c$  は  $0 \leq c \leq n-1$  となる整数である.

- (2) 受信者 B は自分だけが知っている秘密の鍵  $K_D$  を用いて

$$\text{復号} : m = c^d \pmod{n} \quad (9.9)$$

を計算し, 平文  $m$  を得る.

**[定理 9.2]** 式 (9.8), (9.9) において, 式 (9.2) が成り立つ. □

(証明) 式 (9.6) を用いて, ある整数を  $d$  とすると

$$c^d = m^{ed} = m^{a\phi(n)+1} = m \pmod{n} \quad (9.10)$$

<sup>6</sup>通常は  $p-1$  と  $q-1$  の最小公倍数を用いる.

<sup>7</sup>実際には,  $p-1$  と  $q-1$  に対し互いに素であれば十分である.

<sup>8</sup>これは, ユークリッド互除法を用いて求めることができる.

<sup>9</sup> $n$  を公開しているから, 結局  $d$  だけを秘密とする. もちろん,  $p, q$  は秘密である.

<sup>10</sup> $c, (e, n)$  から直接  $m$  を求めることは離散対数問題でやはり困難である.

である<sup>11</sup>. □

#### 9.4 ゼロ (零) 知識証明

**ゼロ知識 (会話型) 証明プロトコル** (Zero Knowledge Interactive Proof : **ZKIP** protocol) は証明者 (認証相手) が持っている秘密情報を全く開示することなく, 認証者 (検証者) にその秘密情報を持っていることを納得させるための通信手順 (プロトコル) である. 次に, A.Fiat と A.Shamir による認証法を示す.

まず, 信頼できるセンターを仮定する. センターは2つの素数  $p, q$  とその積  $n = pq$  を生成し,  $p, q$  を秘密にする. センターは証明者の ID 名などから識別情報  $u$  を生成し,  $u$  から  $v^2 = u \pmod{n}$  となる  $v$  を作り証明者に配布する. この  $v$  が証明者しか知らない**秘密情報**,  $u$  は**公開情報**である. センター以外は  $p, q$  を知らないから  $v$  を求めることは困難である<sup>12</sup>.

証明者だけが  $v$  を持っていることを図 9.2 の手順にしたがって証明する. 図のプロトコルを1回とし, これを  $t$  回実行する. 証明者が  $v$  を持たないにもかかわらず, このテストにすべて合格する確率は  $2^{-t}$  となる.

この手順により

- ① (完全性)  $v$  を知っている証明者のみが検証者を納得させることができる.
- ② (健全性) 証明者に代ってなりすましができる確率は  $2^{-t}$  である.
- ③ (ゼロ知識性) どのように  $e$  を選んでも  $v$  はわからない.

が成立する.

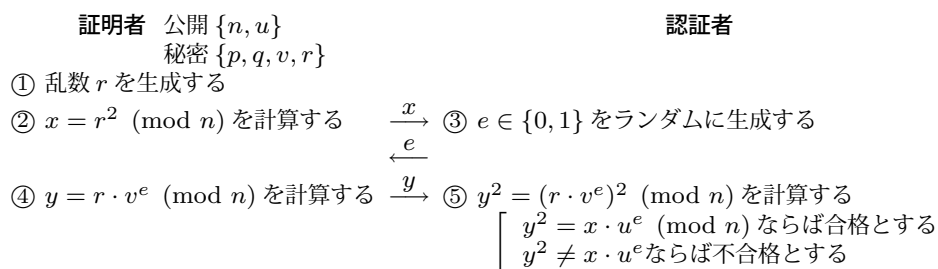


図 9.2: Fiat-Shamir の認証方式

<sup>11</sup>式 (9.10) は整数  $n = pq$  で割った余りの整数の集合  $\mathcal{Z}_n$  上で任意の  $m \in \mathcal{Z}_n$  に対し  $m^{\phi(n)} = 1 \pmod{n}$  であるという**オイラーの定理**を拡張した結果を用いて導かれる. 式 (9.6) より  $ed = a(p-1)(q-1) + 1$  と表わせば, **フェルマーの小定理**  $m^{p-1} = 1 \pmod{p}$  を用いてもよい.

<sup>12</sup>平方剰余問題が NP 完全であることに根拠をおいている.

## 演習問題

- [9.1] RSA 暗号において  $n(=pq)$  の値をユーザによらず共通とすると、安全性はどうなるか。ただし、 $p, q$  は秘密にしておく。
- [9.2] RSA 暗号で鍵  $e$  の値によらず平文  $m$  と暗号文  $c$  が一致する場合を挙げよ。
- [9.3] RSA 暗号で素因数  $p, q$  が容易に求まるような暗号文はどんな場合か。

## 参考文献

1. 池野信一, 小山謙二, 現代暗号理論, 電子通信学会, コロナ社, 昭61年.
2. 辻井重男, 笠原正雄編, 暗号と情報セキュリティ, 昭晃堂, 1990年.
3. 菊地豊, 情報セキュリティ概論, コロナ社, 1994年.
4. 大田和夫, 黒澤馨, 渡辺治, 情報セキュリティの科学, マジックプロトコルへの招待, ブルーボックス, 1995年.
5. 岡本栄司, 暗号理論入門, 共立出版株式会社, 1993年.
6. 情報理論とその応用学会編, 暗号と認証, 情報理論とその応用シリーズ4, 培風館, 1996年.
7. 辻井重男, 暗号, 講談社選書メチエ73, 1996年.
8. ウォーウィックフォード, マイケルバウム, (山田慎一郎訳), デジタル署名と暗号技術, 安全な電子商取引のためのセキュリティシステムと法律基盤, 株式会社ピアソンエデュケーション, 1997年.
9. 今井秀樹, 明るい暗号の話, ネットワーク社会のセキュリティ技術, ポピュラーサイエンス, 裳華房, 1998年.
10. 岡本龍明, 暗号と情報セキュリティ, 情報ネット社会のインフラ技術, 日経BP社, 1998年.
11. 吹田智章, 暗号のすべてが分かる本, デジタル時代の暗号革命, 技術評論社, 1998年.
12. カーライルアダムス, スティーブロイド, (鈴木優一訳), PKI, 公開鍵インフラストラクチャの概念, 標準, 展開, 株式会社ピアソンエデュケーション, 1999年.
13. 今井秀樹, 松浦幹太, 情報セキュリティ概念, 情報セキュリティシリーズ第6巻, 昭晃堂, 1999年.
14. 井上能行, 電子決済システムのしくみ, 入門eビジネス, 日本実業出版社, 2000年.
15. 小松尚久, "バイオメトリック個人認証技術の現状と課題," 信学技報 CQ2000-23, TM2000-21, 2000-07.
16. 中尾康二, "電子商取引のための認証局," 情報処理, vol. 41, no. 3, pp. 282-285, 2000-3.
17. 岡本栄司, 菅知之, エレクトロニックコマース, 情報セキュリティシリーズ第3巻, 昭晃堂, 2000年.
18. 佐々木良一, 吉浦裕, 手塚悟, 三島久典, インターネット時代の情報セキュリティ, 暗号と電子透かし, 共立出版株式会社, 2000年.

## 演習問題解答

## 第1章

[1.1]

$$\begin{aligned}
(x+y) + \bar{x} \cdot \bar{y} &= ((x+y) + \bar{x}) \cdot ((x+y) + \bar{y}) && \text{(分配則)} \\
&= ((x+\bar{x}) + y) \cdot (x + (y+\bar{y})) && \text{(交換則)} \\
&= (1+y)(x+1) && \text{(補元)} \\
&= 1
\end{aligned}$$

( $\because 1+x = x+\bar{x}+x = x+x+\bar{x} = x+\bar{x} = 1, \because x+x = (x+x)1 = (x+x) \cdot (x+\bar{x}) = x+x \cdot \bar{x} = x+0 = x$ )

$$\begin{aligned}
(x+y)(\bar{x} \cdot \bar{y}) &= x(\bar{x} \cdot \bar{y}) + y(\bar{x} \cdot \bar{y}) && \text{(分配則)} \\
&= (x \cdot \bar{x}) \cdot \bar{y} + \bar{x} \cdot (y \cdot \bar{y}) && \text{(交換則)} \\
&= 0 \cdot \bar{y} + \bar{x} \cdot 0 && \text{(補元)} \\
&= 0
\end{aligned}$$

$$a + b = 1, a \cdot b = 0 \iff b = \bar{a}$$

より

$$\overline{x+y} = \bar{x} \cdot \bar{y}.$$

$\overline{x \cdot y} = \bar{x} + \bar{y}$  も同様.

[1.2] 任意のブール関数は主加法標準形式 (1.16), (1.17) で表現できる. これの + と  $\cdot$ , 0 と 1 を入れかえると

$$f^*(x_1, x_2, \dots, x_n) = \prod \left[ \overline{f(a_1, a_2, \dots, a_n)} + x_1^{a_1} + x_2^{a_2} + \dots + x_n^{a_n} \right] \quad (\text{a.1.2.1})$$

である. 一方, 式 (q.1.2.1) からドモルガンの定理を適用して

$$f^*(x_1, x_2, \dots, x_n) = \sum \overline{f(a_1, a_2, \dots, a_n) x_1^{a_1} x_2^{a_2} \dots x_n^{a_n}} \quad (\text{a.1.2.2})$$

$$= \prod \left[ \overline{f(a_1, a_2, \dots, a_n)} + x_1^{a_1} + x_2^{a_2} + \dots + x_n^{a_n} \right] \quad (\text{a.1.2.3})$$

となり式 (a.1.2.1) に等しい.

[1.3] 2変数  $x_1, x_2$  について示せば, 容易に  $n$  変数 ( $n \geq 3$ ) に拡張できる.

- (1) **NAND** :  $\{\bar{\cdot}, \cdot\}$   $z = \overline{x_1 \cdot x_2} = x_1 | x_2$  (**Sheffer の縦棒**)
- 和  $x_1 + x_2 = \overline{(\overline{x_1 \cdot x_1}) \cdot (\overline{x_2 \cdot x_2})} = (x_1 | x_1) | (x_2 | x_2)$
- 積  $x_1 \cdot x_2 = \overline{(\overline{x_1 \cdot x_2}) \cdot (\overline{x_1 \cdot x_2})} = (x_1 | x_2) | (x_1 | x_2)$
- 否定  $\bar{x} = \overline{x \cdot x} = x | x$
- (2) **NOR** :  $\{\bar{\cdot}, +\}$   $z = \overline{x_1 + x_2} = x_1 || x_2$  (**二重縦棒, Pierce の演算**)
- 和  $x_1 + x_2 = \overline{\overline{x_1 + x_2} + \overline{x_1 + x_2}} = (x_1 || x_2) || (x_1 || x_2)$
- 積  $x_1 \cdot x_2 = \overline{\overline{x_1 + x_1} + \overline{x_2 + x_2}} = (x_1 || x_1) || (x_2 || x_2)$
- 否定  $x = \overline{x + x} = x || x$
- (3) **inhibit** :  $\{/, 1\}$   $z = x_1 \cdot \bar{x}_2 = x_1 / x_2$
- 和  $x_1 + x_2 = 1 / ((1 \cdot \bar{x}_1) \cdot \bar{x}_2) = 1 / ((1/x_1) / x_2)$
- 積  $x_1 \cdot x_2 = x_1 \cdot (\overline{1 \cdot \bar{x}_2}) = x_1 / (1/x_2)$
- 否定  $\bar{x} = 1 \cdot \bar{x} = 1/x$

第2章

[2.1] 個体定義領域  $\mathcal{D}$  において,  $\exists x P(x)$  は  $P(d) = 1$  となる  $d \in \mathcal{D}$  が存在するとき 1, そうでないとき 0 である. したがって, 与えられた論理式が恒真式でないとするとき, この論理式を 0 とするような解釈  $\mathcal{D}$  が存在する. この仮定より  $\exists x \forall y F(x, y) = 1$ , かつ  $\forall y \exists x F(x, y) = 0$  となる  $F(x, y)$  に対する  $\mathcal{D}$  が存在する. このとき,  $d \in \mathcal{D}$  と任意の  $d' \in \mathcal{D}$  に対し  $F(d, d') = 1$  となる  $d$  が存在する. これは  $\forall y \exists x F(x, y) = 0$  に矛盾する. したがって, 恒真式である.

たとえば,  $x, y$  を自然数とし,  $F(x, y)$  を  $x > y$  となる集合とすると,  $\forall y \exists x (x > y)$  は任意の自然数  $n$  に対し  $y = n$  とすると  $\exists x (x > n)$  が正しいことを示す.  $x = n + 1$  とおけば,  $x > y$  より明らか. 一方,  $\exists x \forall y (x > y)$  は,  $\forall y (n > y)$  となる  $x = n$  が存在することを要求している. しかしすべての自然数より大きな  $n$  は存在しない.

[2.2] (1) 
$$\frac{\frac{A \quad A \supset B}{B} \quad \frac{A \quad A \supset (B \supset C)}{B \supset C}}{C} \quad \frac{C}{A \supset C}$$

(2)  $A \wedge B$  は  $A$  と  $B$  の両方が成立するとき, かつそのときに限って成立することを示す. したがって

$$\frac{A \quad B}{A \wedge B}, \quad \frac{A \wedge B}{A}, \quad \frac{A \wedge B}{B}$$

は推論規則に従うことに注意する.

$$\frac{\frac{A \wedge B}{B} \quad \frac{\frac{A \wedge B}{A} \quad A \supset (B \supset C)}{B \supset C}}{C} \quad \frac{C}{(A \wedge B) \supset C} \quad \frac{(A \wedge B) \supset C}{(A \supset (B \supset C)) \supset ((A \wedge B) \supset C)}$$

[2.3] 素数  $x$  は  $x > 1$  かつ 1 とそれ自身以外に約数  $y$  をもたない自然数であるから

$$(x > 1) \wedge \forall y[\exists z(x = yz) \supset (y = 1 \vee y = x)]$$

[2.4]  $p \supset q \equiv \neg p \vee q \equiv \neg(p \wedge \neg q)$  が恒真, すなわち  $q$  の否定  $\neg q$  を前提に加えて,  $p \wedge \neg q$  が恒偽であることを示す. したがって,  $q$  を否定すれば矛盾が導かれる. 例えば,  $p \supset q, r \supset s, s \supset \neg q$  を前提とし  $r \supset \neg p$  を導くには  $(p \supset q) \wedge (r \supset s) \wedge (s \supset \neg q) \wedge \neg(r \supset \neg p)$  が偽であることを示せばよい.

### 第3章

[3.1] 入力系列 (1), (2) のオートマトンの遷移図をそれぞれ図 a.3.1.1 (1),(2) に示す.

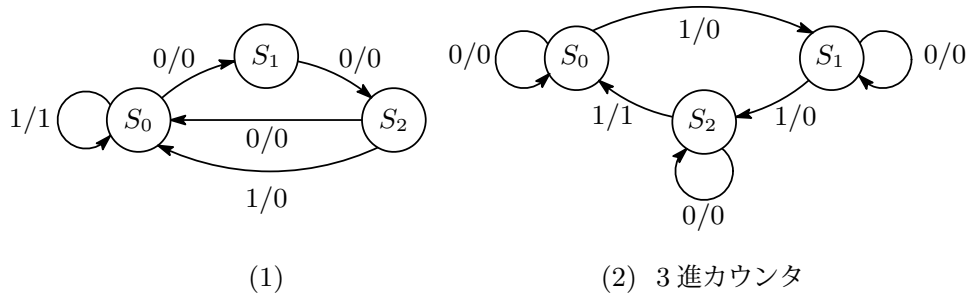


図 a.3.1.1: オートマトンの遷移図

[3.2] 生成規則を 3.2 節のように  $\xRightarrow{G}$  で示す (以後  $G$  は省略する).  $S$  は開始記号,  $X, Y, Z$  は非終端記号 (中間記号),  $|$  は接続を示す.

$$\begin{aligned} S &\Longrightarrow X \\ X &\Longrightarrow (|Y|) \\ X &\Longrightarrow Y| + |Z \\ X &\Longrightarrow Y| \times |Z \\ Y &\Longrightarrow X \\ Z &\Longrightarrow X \\ X, Y, Z &\Longrightarrow a, b, c \end{aligned}$$

### 第4章

[4.1]  $T = \{1, 0, S, E, \alpha\}, a = (S0^n 1^n E)$  とする.

- ① まず  $S$  を読む. 右へ移動する.
- ② 1 を読むまで 0 を読みとばし, 右へ移動しつづける.
- ③ 1 を読むとこれを  $\alpha$  にかえて左に移動する.
- ④ 0 を読むとこれを  $\alpha$  にかえて右に移動する. これを 0,1 がすべて  $\alpha$  にかわるまで続ける.
- ⑤  $E$  を読み左へ移動し,  $\alpha$  を読んで  $S$  を読めば受理する. それ以外のときは受理しない.

$$[4.2] \quad (1) \quad \begin{cases} f(0, y) = y \\ f(x+1, y) = f(x, y) + 1 \end{cases}$$

詳しくは次の通り.

$$\begin{aligned} \varphi_1(x) &= x + 1 \\ \varphi_2(b) &= b \\ \varphi_3(a, x, b) &= x \\ \varphi_4(a, x, b) &= \varphi_1(\varphi_3(a, x, b)) \\ \varphi_5(0, b) &= \varphi_2(b) = b \\ \varphi_5(a+1, b) &= \varphi_4(a, \varphi_5(a, b), b) \\ &= \varphi_1(\varphi_3(a, \varphi_5(a, b), b)) \\ &= \varphi_1(\varphi_5(a, b)) = \varphi_5(a, b) + 1 \end{aligned}$$

よって,  $f(a, b) = \varphi_5(a, b)$  となる.

$$(2) \quad \begin{aligned} f(0) &= 1 \\ f(x+1) &= (x+1)f(x). \end{aligned}$$

## 第5章

[5.1] (1)  $P_{\text{GCD}} : M, N (M \geq N) \in \mathbb{Z}_+ \rightarrow \text{gcd} \in \mathbb{Z}_+$  である.

(i) 数式表現

$a_0 = m, a_1 = n$  とし

$$a_0 = q_1 a_1 + a_2 \quad (a_1 > a_2)$$

$$a_1 = q_2 a_2 + a_3 \quad (a_2 > a_3)$$

$\vdots$

$$a_{i-1} = q_i a_i + a_{i+1} \quad (a_i > a_{i+1})$$

$$a_i = q_{i+1} a_{i+1}, a_{i+2} = 0$$

のとき

$$\text{gcd} = a_{i+1}$$

(ii) 箇条書き

(a) (初期化)  $i = 0, a_0 = M, a_1 = N$  とおく

(b) (繰返し)  $a_i = q_{i+1} a_{i+1} + a_{i+2} (a_{i+1} > a_{i+2})$  を計算する.

(c) (判定・ジャンプ)  $a_{i+2} = 0$  ならば  $\text{gcd} = a_{i+1}$  とし停止,  $a_{i+2} > 0$  ならば  $i = i + 1$  として2へ

(iii) フローチャート

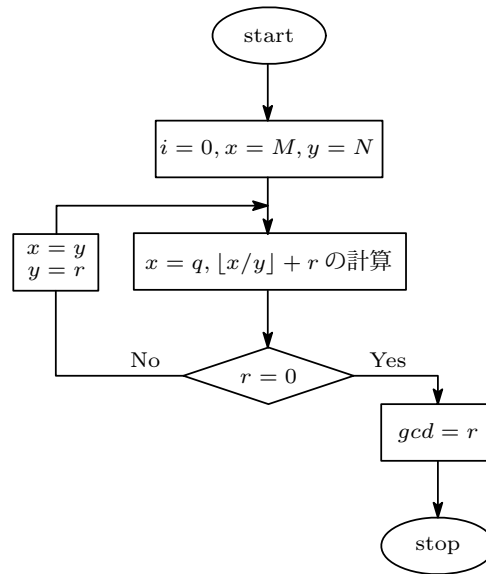


図 a.5.1.1: フローチャート

## (iv) ステップ記述

ステップ 1 (初期化)  $x = M, y = N$

ステップ 2 (繰返し)  $r = x - y[x/y]$

ステップ 3 (判定・ジャンプ)  $r = 0$  ならば  $gcd = y$  とし停止,  $r > 0$  ならば  $x \leftarrow y, y \leftarrow r$  としステップ 2 へ

## (v) 擬似的 PASCAL

```

function gcd(m,n)
begin
  x に m を, y に n を代入する.
  while y ≠ 0 do
  begin
    x を y で割ったときの余りを r とする.
    x に y を, y に r を代入する.
  end
  gcd=x とする
end
end
  
```

図 a.5.1.2: プログラム (1)

## (2) プログラム

```

1 function gcd(m,n: integer):integer; (m>n)
2 var x,y,q,r: integer;
3 begin
4   x:=m; y:=n;
  
```



```

5   while y ≠ 0 do
6     begin
7       q:=x div y; r:=x-y*q;
8       x:=y; y:=r;
9     end;
10  gcd:=x;
11 end;

```

図 a.5.1.3: プログラム (2)

(プログラム ... 再帰型の場合)

```

1 function gcd(m,n: integer):integer;
2 begin
3   if n=0 then gcd:=m
4     else gcd:=gcd(n,m mod n)
5 end;

```

図 a.5.1.4: プログラム (3)

(3) 図 a.5.1.1 フローチャートの繰り返し部分 (図 a.5.1.3 の 5 行目 `while` から 9 行目 `end`) の回数よりその上界は  $O(\log m)$  である。

[5.2] 問題  $A$  が NP 困難であれば, 問題  $A$  に多項式還元可能な問題  $B \in \mathcal{NP}$  が存在する. もし  $A \in \mathcal{P}$  ならば,  $B \in \mathcal{P}$ ,  $A \notin \mathcal{P}$  ならば  $B \notin \mathcal{P}$  であるから,  $\mathcal{P} \supseteq \mathcal{NP}$ . よって, 式 (5.3) より  $\mathcal{P} = \mathcal{NP}$ .

## 第 7 章

[7.1]  $H(\mathcal{A}) = 2.45$  [bits/symbol]

$$\bar{l} = 2(0.3 + 0.2) + 3(0.16 + 0.15 + 0.12 + 0.07) = 2.5$$

である. 明らかに,  $H(\mathcal{A}) \leq \bar{l}$  である.

[7.2]  $H(\mathcal{A}) = 1.75$  [bits/symbol],  $\bar{l} = 1 \times 0.5 + 2 \times 0.25 + 3(0.125 + 0.125) = 1.75$ .

このとき,  $H(\mathcal{A}) = \bar{l}$  が成り立つ.

[7.3]  $f(t)$  のフーリエ変換  $F(\omega)$  は

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt = 0 \quad (|\omega| > \omega_0) \quad (\text{a.7.3.1})$$

である.  $F(\omega)$  は  $|\omega| < \omega_0$  でのみ 0 でない値をとるから,  $F(\omega)$  はフーリエ級数で展開でき

$$F(\omega) = \sum_{m=-\infty}^{\infty} C_m e^{-j\left(\frac{m\pi}{\omega_0}\right)\omega} \quad (|\omega| \leq \omega_0) \quad (\text{a.7.3.2})$$

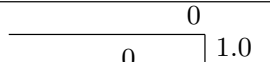
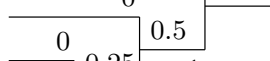
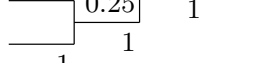
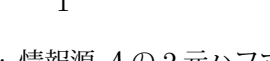
$\mathcal{A}$	$p_i$		符号語 $\mathbf{v}_{(i)}$
$a_1$	0.5		0
$a_2$	0.25		10
$a_3$	0.125		110
$a_4$	0.125		111

図 a.7.2.1: 情報源  $\mathcal{A}$  の 2 元ハフマン符号

ただし

$$C_m = \frac{1}{2\omega_0} \int_{-\omega_0}^{\omega_0} F(\omega) e^{j\left(\frac{m\pi}{\omega_0}\right)\omega} d\omega \quad (\text{a.7.3.3})$$

である。このとき、逆変換を求めると

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega \quad (\text{a.7.3.4})$$

であるから、 $t = \frac{m\pi}{\omega_0}$  において

$$f\left(\frac{m\pi}{\omega_0}\right) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\left(\frac{m\pi}{\omega_0}\right)\omega} d\omega \quad (\text{a.7.3.5})$$

である。これと式 (a.7.3.3) を比較すると

$$f\left(\frac{m\pi}{\omega_0}\right) = \frac{\omega_0}{\pi} C_m$$

となる。その結果、式 (a.7.3.3) を用いて帯域制限された連続関数  $f(t)$  は  $\frac{m\pi}{\omega_0}$  ( $m = -\infty, \dots, -1, 0, 1, 2, \dots, +\infty$ ) の値、すなわち  $T_0 = \frac{\pi}{\omega_0} = \frac{\pi}{2\pi f_0} = \frac{1}{2f_0}$  ごとの標本値で完全に定まる。以上より

$$\begin{aligned} f(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega \\ &= \frac{1}{2\pi} \int_{-\omega_0}^{\omega_0} \left[ \sum_{m=-\infty}^{\infty} C_m e^{-j\left(\frac{m\pi}{\omega_0}\right)\omega} \right] e^{j\omega t} d\omega \\ &= \frac{\omega_0}{\pi} \sum_{m=-\infty}^{\infty} C_m \frac{\sin \omega_0 \left(t - \frac{m\pi}{\omega_0}\right)}{\omega_0 \left(t - \frac{m\pi}{\omega_0}\right)} \\ &= \sum_{m=-\infty}^{\infty} f\left(\frac{m\pi}{\omega_0}\right) \frac{\sin \omega_0 \left(t - \frac{m\pi}{\omega_0}\right)}{\omega_0 \left(t - \frac{m\pi}{\omega_0}\right)} \end{aligned}$$

となり、式 (7.11) が得られる。

[7.4] (1)  $\forall x > 0$  に対し

$$\log_e x \leq x - 1 \quad (x > 0) \quad (\text{a.7.4.1})$$

$x = q_i/p_i$  とすると

$$\sum p_i \log_2(q_i/p_i) \leq \sum (q_i - p_i)/\log_e 2 = 0 \quad (\text{a.7.4.2})$$

式 (a.7.4.1) の等号は  $x = 1$  のとき成り立つ。

(2) 式 (a.7.4.1) において,  $I(\mathbf{p}||\mathbf{q}) \geq 0$  より  $q_i = 1/n$  とおくと,

$$H(\mathcal{A}) \leq -\sum p_i \log(1/n) = \log n \quad (\text{a.7.4.3})$$

## 第 8 章

[8.1] (1)  $7 \times 2 + 9 \times 3 + 8 \times 4 + 0 \times 5 + 4 \times 6 + 5 \times 7 + 2 \times 1 = 134 = 2 \pmod{11}$ , したがって正しくない。

(2)  $7 \times 2 + 9 \times 3 + 8 \times 4 + 0 \times 5 + X \times 6 + 3 \times 7 + 6 \times 1 = 100 + 6X = 0 \pmod{11}$  を解いて  $X = 9$ 。

(3)  $7 \times 2 + 7 \times 3 + 8 \times 4 + 0 \times 5 + 7 \times 6 + 1 \times 7 + 0 \times 1 = 116 \neq 0 \pmod{11}$ , したがって入れかえ誤りを発見できる<sup>13</sup>。

[8.2]

$$GH^T = [I_k, P] \begin{bmatrix} -P \\ I_{n-k} \end{bmatrix} = [P - P] = 0 \pmod{2}$$

である。

[8.3] (1)

$$H = \begin{bmatrix} 1101100 \\ 1011010 \\ 0111001 \end{bmatrix} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_7], \quad G = \begin{bmatrix} 1000110 \\ 0100101 \\ 0010011 \\ 0001111 \end{bmatrix}$$

(2) (1) の  $G$  に対し,

$$\mathbf{x} = \mathbf{w}G = (0101010)$$

(3)  $\mathbf{y}_1 = \mathbf{x} + \mathbf{e}_1 = (0100010)$

$$\mathbf{y}_2 = \mathbf{x} + \mathbf{e}_2 = (0111011)$$

$$\mathbf{s}_1 = \mathbf{y}_1 H^T = (111)$$

$$\mathbf{s}_2 = \mathbf{y}_2 H^T = (010)$$

(4)  $\mathbf{s}_1^T = \mathbf{h}_4$  より  $\hat{\mathbf{e}} = (0001000) = \mathbf{e}_1$  で正しく訂正する。  $\mathbf{s}_2^T = \mathbf{h}_6$  より,  $\hat{\mathbf{e}} = (0000010) \neq \mathbf{e}_2$  で誤まった訂正 (誤訂正) をする。

## 第 9 章

[9.1] ユーザ  $i$  の暗号化鍵を  $e_i$ , 復号鍵を  $d_i$  とする。あるユーザがユーザ  $i, j$  に同一の平文  $m$  を送るとする。もし, 暗号化鍵  $e_i$  と  $e_j$  が互いに素ならば,  $ae_i + be_j = 1$  となる  $a, b$  が存在する。したがって, ユーザ  $i, j$  への暗号文  $c_i, c_j$  は

$$c_i = m^{e_i} \pmod{n}$$

$$c_j = m^{e_j} \pmod{n}$$

<sup>13</sup> $x_7 = 0$  のとき, 計算結果が 10 の場合と 0 の場合があることに注意。このケースでは 10 においても誤りが発見できる。

であるから

$$c_i^a \cdot c_j^b = (m^{e_i})^a (m^{e_j})^b = m^{ae_i+be_j} = m \pmod{n}$$

となり, 平文  $m$  が露呈してしまう. 盗聴者は  $e_i, e_j, n$  を知っているから,  $\text{GCD}(e_i, e_j) = 1$  のときの同報  $m$  の安全性は保証されない.

(注) ユーザ  $i$  は自分の鍵の対  $(e_i, d_i)$  と  $n$  を知っている.  $p, q$  は知らないが, ユーザ  $j$  の暗号化鍵  $e_j$  は公開されているから,  $n$  が共通ならば,

$$e_j d_j = 1 \pmod{\phi(n)}$$

ただし

$$\exists k, k\phi(n) = e_i d_i - 1$$

として  $d_j$  を計算できる可能性がある.

[9.2]  $e$  は奇数とする.  $n$  の値によらず

$$c = m^e = m \pmod{n}$$

- となる場合は
- ①  $m = 0 \pmod{p}$  かつ  $m = 0 \pmod{q}$
  - ②  $m = 0 \pmod{p}$  かつ  $m = 1 \pmod{q}$
  - ③  $m = 0 \pmod{p}$  かつ  $m = -1 \pmod{q}$
  - ④  $m = 1 \pmod{p}$  かつ  $m = 0 \pmod{q}$
  - ⑤  $m = 1 \pmod{p}$  かつ  $m = 1 \pmod{q}$
  - ⑥  $m = 1 \pmod{p}$  かつ  $m = -1 \pmod{q}$
  - ⑦  $m = -1 \pmod{p}$  かつ  $m = 0 \pmod{q}$
  - ⑧  $m = -1 \pmod{p}$  かつ  $m = 1 \pmod{q}$
  - ⑨  $m = -1 \pmod{p}$  かつ  $m = -1 \pmod{q}$

[9.3]  $c \neq 0 \pmod{n}$  であるか  $c = 0 \pmod{p}$  (または  $c = 0 \pmod{q}$ ) のとき  $c \neq 0 \pmod{n}$  ) かつ  $\text{GCD}(c, e) = g \neq 1$  ならば  $g = p$ , したがって  $q = n/g$  (または  $g = q$ , したがって  $p = n/g$ ) となる. すなわち,  $c = 0 \pmod{p}$  かつ  $c \neq 0 \pmod{q}$  (または  $c \neq 0 \pmod{q}$  かつ  $c = 0 \pmod{q}$ ) の場合である.