

数学的帰納法 mathematical induction

- ◆ ペアノ Peano の公理： 次の公理によって規定される集合 \mathbb{N} の要素を **自然数** という。
 - $0 \in \mathbb{N}$
 - $x \in \mathbb{N}$ ならば $S(x) \in \mathbb{N}$
 - $S(x)=S(y)$ ならば $x=y$ (単射)
 - $x \in \mathbb{N}$ ならば $S(x) \neq 0$
 - 集合 M が $0 \in M$ および $(x \in M \Rightarrow S(x) \in M)$ を満たすならば、 $\mathbb{N} = M$
- ◆ 自然数の例：

$$0, S(0), S(S(0)), S(S(S(0))), S(S(S(S(0)))) \dots$$

公理 (v) 数学的帰納法

- ◆ $P(n)$ を自然数 n に関する述語とする。
 $P(n)$ を満たす自然数 n の集合を M とする。

- $P(0)$
- $\forall k \in M (P(k) \Rightarrow P(S(k)))$

- ◆ 上の a と b を証明できれば、 $\mathbb{N} = M$

$$\forall n \in \mathbb{N} (P(n))$$

この証明法が**数学的帰納法** (mathematical induction) 0ではなく、 n (任意の $n \in \mathbb{N}$) からスタートして良い。

数学的帰納法による証明の例

$$\forall n \left\{ 1+2+3+4+\dots+n = \frac{n(n+1)}{2} \right\}$$

- (a) 帰納法の**基礎** (base, basis) $0 = \frac{0(0+1)}{2}$

- (b) 帰納法の**ステップ** (step)

$$1+2+3+4+\dots+k = \frac{k(k+1)}{2} \text{ が正しいと仮定すると}$$

$$1+2+\dots+k+(k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}$$

帰納法のステップにおける k は任意の自然数であった。
 (a)と(b)により、**帰納法**で証明された。

累積帰納法

- ◆ $P(n)$ を自然数 n に関する性質 (述語) とする。

- $P(0)$
- $\forall k \{ (P(0) \wedge P(1) \wedge \dots \wedge P(k)) \Rightarrow P(S(k)) \}$

a と b から下を導く。

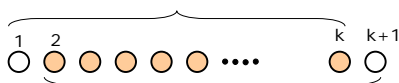
$$\forall n (P(n))$$

注意： $P(0) \wedge P(1) \wedge \dots \wedge P(k)$ は step 中の**仮定** として使うものであり、正しいとは限らない。

間違った証明の例

ヒント: $k=1$ の場合を考えよ。

- ◆ すべての自然数 $1 \leq n$ について n 人の集団は同性から成る。
 - $n=1$ のとき、その1人が女性であっても男性であっても正しい。
 - $k+1$ 人の集団は、1人と k 人から成る。帰納法の**仮定**により、どの k 人以下の集団も同性。



よって $k+1$ 人の集団も同性より成る。証明終。

帰納的定義、再帰的定義

- ◆ 集合、関数、性質、関係などを次のように三段階に定義する。関数の**集合 F**を定義する場合を例示。
 - 初期値**：最初から集合 F に含まれる関数を列挙する。
 - 再帰**：既に F に含まれると分っている関数から、 F に含まれる新しい関数を構成する方法を述べる。
 - 限定**：上の a と b によって構成されるものだけが集合 F の要素 (元) であると限定する。
- このような定義の方法を**帰納的** (inductive) 定義、あるいは**再帰的** (recursive) 定義という。

7

原始帰納的関数 (i) ~ (iv)

- i. 関数 $f(x_1, x_2, \dots, x_n) = k$ は原始帰納的関数である。ここに $k \in \mathbb{N}$ (自然数)。 **定数関数**
- ii. 関数 $f(x) = S(x)$ は原始帰納的関数である。 **後者関数**
- iii. 関数 $U_i^n(x_1, x_2, \dots, x_n) = x_i$ ($1 \leq i \leq n$) は原始帰納的関数である。 **射影関数**
- iv. $g(x_1, x_2, \dots, x_m), h_1(x_1, x_2, \dots, x_n), h_2(x_1, x_2, \dots, x_n), \dots, h_m(x_1, x_2, \dots, x_n)$ が原始帰納的関数であるとき、次のように定義される関数 f は原始帰納的関数。
 $f(x_1, x_2, \dots, x_n) =$ **合成・代入**
 $g(h_1(x_1, x_2, \dots, x_n), h_2(x_1, x_2, \dots, x_n), \dots, h_m(x_1, x_2, \dots, x_n))$

8

原始帰納的関数 (v_a), (v_b) 原始帰納法

- v_a . $h(x_1, x_2)$ が原始帰納的関数で、 $k \in \mathbb{N}$ のとき、次のように定義される関数 $f(x)$ は原始帰納的関数。
 $f(0) = k$
 $f(S(x)) = h(f(x), x)$
- v_b . $g(x_1, x_2, \dots, x_n), h(x_1, x_2, \dots, x_{n+2})$ がともに原始帰納的関数であるとき、次のように定義される関数は原始帰納的関数。
 $f(0, x_1, x_2, \dots, x_n) = g(x_1, x_2, \dots, x_n)$
 $f(S(x), x_1, x_2, \dots, x_n) = h(f(x, x_1, x_2, \dots, x_n), x, x_1, x_2, \dots, x_n)$

◆ 上の(i)から(v)までによって原始帰納的関数であるとわかるものだけが原始帰納的関数である。

9

帰納的定義と原始帰納法

- ◆ f の定義が他の関数に依存してもよい
 - a. $f(0, y) = g(y)$
 - b. $f(S(x), y) = h(f(x, y), x, y)$ (g, h はすでに定義された関数)
- ◆ 例: (i) ~ (v)
 - $plus(0, y) = y$
 - $plus(S(x), y) = S(plus(x, y))$
 - $times(0, y) = 0$
 - $times(S(x), y) = plus(times(x, y), y)$

10

原始帰納的関数 (続)

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

- $factorial(0) = 1$
 $factorial(S(x)) = g(x, factorial(x))$
 where $g(x, z) = times(S(x), z)$
- $pd(0) = 0$
 $pd(S(x)) = x$
- ◆ フィボナッチ Fibonacci 関数の下記の定義は原始帰納的でないが原始帰納的関数である。
 - $fib(0) = 0$
 - $fib(S(0)) = S(0)$
 - $fib(S(S(x))) = plus(fib(S(x)), fib(x))$

11

二重帰納法

- ◆ 二つの自然数 x, y に関する帰納法
- ◆ 例: アッカーマン (Ackermann) 関数
 - $A(0, y) = y + 1$
 - $A(x + 1, 0) = A(x, 1)$
 - $A(x + 1, y + 1) = A(x, A(x + 1, y))$
- この関数はプログラムで記述することができる。
- ◆ Ackermann 関数は原始帰納的関数ではない。Ackermann 関数は帰納的関数である。

12

アッカーマン Ackermann 関数

- $A(0, y) = y + 1$
- $A(x + 1, 0) = A(x, 1)$
- $A(x + 1, y + 1) = A(x, A(x + 1, y))$
- $A(1, 0) = A(0, 1) = 2$
- $A(1, y + 1) = A(0, A(1, y)) = A(1, y) + 1$
- $A(2, 0) = A(1, 1) = 3$
- $A(2, y + 1) = A(1, A(2, y)) = A(2, y) + 2$
- $A(3, 0) = A(2, 1) = 5$
- $A(3, y + 1) = A(2, f(3, y)) = 2A(3, y) + 3$

x を固定すれば原始帰納的関数 $A_n(y)$

- $A(1, y) = (2 + y + 3) - 3$
- $A_1(y) = y + 2$
- $A(2, y) = 2(y + 3) - 3$
- $A_2(y) = 2y + 3$
- $A(3, y) = 2^{y+3} - 3$
- $A_3(y) = 2^{y+3} - 3$

13

Ackermann 関数は原始帰納的ではない

- ◆ 第1引数 x を固定すると $A_n(y)$ は y の原始帰納的関数になる。
- ◆ ただし任意の原始帰納的関数 f に対して、すべての x_1, x_2, \dots, x_n に対して次の式が成立つような定数 c が存在する。

$$f(x_1, x_2, \dots, x_n) < A(c, \max(x_1, x_2, \dots, x_n))$$
- ◆ もしアッカーマン関数が原始帰納的であるとすると矛盾が導かれる。もし $A(x, y)$ が原始帰納的関数であるとすると $f(y) = A(y, y)$ も原始帰納的関数である。定数 c が存在して $f(y) < A(c, y)$ となる。ここで y に c を代入すると $f(c) = A(c, c) < A(c, c)$

14

帰納的関数 (原始帰納的関数の拡張)

- ◆ 原始帰納的関数 $f(y)$ もしくは $f(x_1, \dots, x_n, y)$ を考える。
- ◆ どのような x_1, \dots, x_n に対しても $f(x_1, \dots, x_n, y) = 0$ を満たす y が必ず存在するとき、その中で**最小の** y の値を $\mu y.f(x_1, \dots, x_n, y)$ と書く。
- ◆ 原始帰納的関数に加えて、この μ 記法を許して作った関数を**帰納的関数**と呼ぶ。
- ◆ 帰納的関数はチューリングマシンで計算可能である。逆に、チューリングマシンで計算できる関数は帰納的関数として定義できる。

15

帰納的定義、再帰的定義

- ◆ 情報の世界では、自然数以外のものを帰納的に定義することがある。
 - 例：簡単な数式の定義
 - 変数 x は数式である
 - 自然数 $n \in \mathbb{N}$ は数式である
 - E_1 と E_2 が数式するとき、 $E_1 + E_2$ および $E_1 \times E_2$ も数式である
 - E が数式するとき、 (E) も数式である
 - 以上のように作られる**ものだけが数式である**

括弧

16

構造帰納法 structural induction

- ◆ 構造帰納法で定義したものの性質は、その構造に関する数学的帰納法で証明する
 - 数式に出現する左括弧と右括弧の数は等しい
 - 変数については正しい
 - 自然数については正しい
 - E_1 と E_2 が上の性質を満たすとき、 $E_1 + E_2$ および $E_1 \times E_2$ もこの性質を満たす
 - E が上の性質を満たすとき、 (E) もこの性質を満たす

括弧

17

プログラムの解析

- ◆ 少し工夫したべき乗 (a^n) 計算アルゴリズム
 - ◆ $b = 1$
 - ◆ n が正である限り
 - ◆ もし n が奇数なら $b = b \times a$ (偶数ならNOP)
 - ◆ $n = n/2$
 - ◆ $a = a \times a$
 - ◆ b を出力する
- ◆ このアルゴリズムは停止するか
 想定した通りの正しい結果 ($b = a^n$) を返すか

n の最下位ビットが 1
 n を 1 ビット右シフト
 剰余は無視する

18

プログラムの正当性

- ◆ $b = 1$
- ◆ n が正である限り
 - ◆ もし n が奇数なら $b = b \times a$
 - ◆ $n = n/2$
 - ◆ $a = a \times a$
 - ◆ b を出力する

$a_{k+1}, n_{k+1}, b_{k+1}$
 a, n, b の間の
 関係を求める

	a	n	b
最初	3	5	1
1回目	9	2	3
2回目	81	1	3
3回目	6561	0	243

解析

$b_{k+1} = b_k$
 $n_{k+1} = n_k / 2$
 $a_{k+1} = a_k \times a_k$

$b_{k+1} = b_k \times a_k$
 $n_{k+1} = (n_k - 1) / 2$
 $a_{k+1} = a_k \times a_k$

- ◆ $b = 1$
- n が正である限り
 - ◆ n が偶数なら n 偶数 > 0
 - ◆ n が奇数なら n 奇数
 - ◆ もし n が奇数なら $b = b \times a$
 - ◆ $n = n / 2$
 - ◆ $a = a \times a$
- ◆ b を出力する

を繰り返す

	a	n	b
0	3	5	1
1	9	2	3
2	81	1	3
3	6561	0	243

$a_{k+1}^{n_{k+1}} \times b_{k+1} = a_k^{n_k} \times b_k$

invariant

不変式

$b_{k+1} = b_k$
 $n_{k+1} = n_k / 2$
 $a_{k+1} = a_k \times a_k$

$b_{k+1} = b_k \times a_k$
 $n_{k+1} = (n_k - 1) / 2$
 $a_{k+1} = a_k \times a_k$

- ◆ $b = 1$
- n が正である限り
 - ◆ n が偶数なら n 偶数 > 0
 - ◆ n が奇数なら $b = b \times a$
 - ◆ $n = n / 2$
 - ◆ $a = a \times a$
- ◆ b を出力する

を繰り返す

不変式

$a_{k+1}^{n_{k+1}} \times b_{k+1} = a_k^{n_k} \times b_k$

n_k が偶数 > 0

$a_{k+1}^{n_{k+1}} \times b_{k+1} = (a_k \times a_k)^{n_k/2} \times b_k = a_k^{n_k} \times b_k$

n_k が奇数

$a_{k+1}^{n_{k+1}} \times b_{k+1} = (a_k \times a_k)^{(n_k-1)/2} \times (b_k \times a_k) = a_k^{n_k} \times b_k$

21

正当性

- ◆ $b = 1$
- n が正である限り
 - ◆ $n=0$ のとき $b=1=a^0$ 正しい、停止する。
 - ◆ もし n が奇数なら $b = b \times a$
 - ◆ $n = n / 2$
 - ◆ $a = a \times a$
- ◆ b を出力する

を繰り返す

$n > 0$ のとき

$$a_{k+1}^{n_{k+1}} \times b_{k+1} = a_k^{n_k} \times b_k$$

$$= a_0^{n_0} \times b_0$$

$$= a^n \times 1$$

n が k ビットの2進数の時、 $n_k=0$

$$a_k^0 \times b_k = a^n \times 1$$

正しい、停止する。

22

$n_k=0$ の証明

- ◆ 例題 :
 - $n_0 = 111, n_1 = 11, n_2 = 1, n_3 = 0.$
 - $n_0 = 1000, n_1 = 100, n_2 = 10, n_3 = 1, n_4 = 0.$
- ◆ 本来は証明を要する。もしも「証明をせよ」と言われたら、数学的帰納法を使って証明すれば良い。

2004年度定期試験問題 23

演習問題：間違った証明の例 L_kをL₁と表記

- ◆ 平面上に異なる n (≥ 2)本の直線がある。ただし、どの2本も平行ではない。このとき、すべての直線は1点で交わる。
- ◆ 証明
 - (base) $n=2$ のとき：同一平面上の平行でない2直線は1点で交わる。主張は正しい。(a)
 - (step) $n=k$ のときに主張が成り立つと仮定し、 $k+1$ 本の直線 $L_1, L_2, \dots, L(k+1)$ を考える。
 - 帰納法の仮定により、 k 本の直線、 L_1, L_2, \dots, L_k は1点を共有する。その点を x とする。(b)
 - 同様に、 k 本の直線、 $L_1, L_2, \dots, L(k-1), L(k+1)$ も1点を共有する。その点を y とする。(c)

24

演習問題：続き

- L_1 は両方に属しているのだから x も y も通る。(d)
- $L(k-1)$ も両方に属しているのだから x も y も通る。(e)
- L_1 と $L(k-1)$ は1点のみで交わる。 $x=y$ となる。(f)
- したがって $L_1, L_2, \dots, L(k+1)$ は一点 $x(=y)$ で交わる。(g)
- ◆ 問題：以上の証明はどこかで誤っている。どこが、何故誤りなのかを説明せよ。
- ◆ ヒント：先の例題を思い出すこと。 k が小さい場合が怪しい。
- ◆ 解答：(f)が誤り。 $k=2$ のときには L_1 と L_2 とは同一の直線だから、1点のみで交わるとは言えない。